

Número 6 • Abril 2006

Artículo
Compresión Huffman

Entrevista
Wolf

Ya disponible
Ink

¡El ganador de MSX DEV '05!

THE CURE

NÚMERO 6

ABRIL / 2006



Sumario

secciones

Eventos: 28ª RU de Barcelona

La reunión de usuarios de MSX con más historia dentro del panorama nacional volvió a ser la de las grandes ocasiones. Mucha presencia y novedades frescas.





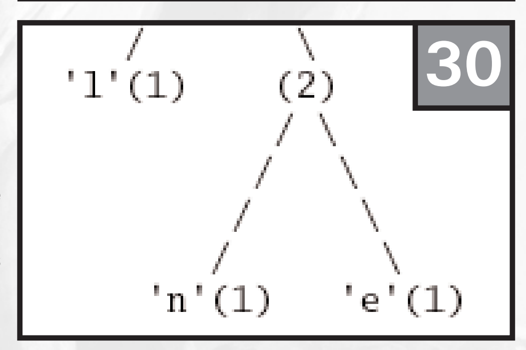
Ya disponible: KONAMI QUIZ 2

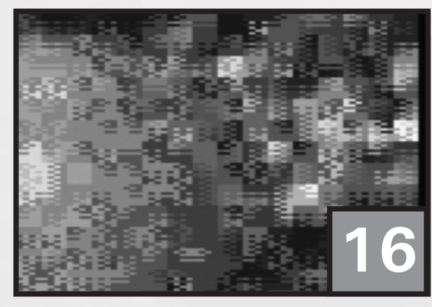
Delta Soft reaparece con un nuevo título inspirado en la casa Konami. Veremos que nos aporta esta secuela como novedad. Además Bloody Paws y el genial Ink.



Artículo: Compresión Huffman

De nuevo Avelino Herrera vuelve a la carga con un artículo muy interesante sobre la compresión de datos para formatos de MSX. Además Ramones nos trae un articulazo de programación en ensamblador.





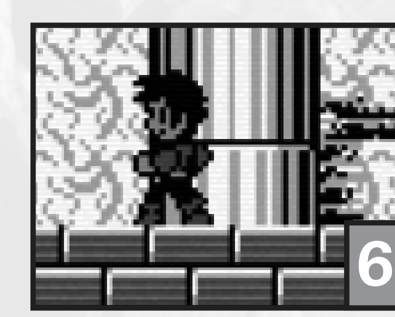
Entrevista: WOLF

Conoce de primera mano quién hay detrás del seudónimo Wolf, quizá el mejor compositor holandés de nuestro sistema.



Software Amateur: MSX Unleashed

La demo de MSX1 que ha superado los límites "imaginables" de la máquina. Pero hay muchos programas más en esta sección.



Mapa fotográfico: THE CURE

El juego ganador del concurso MSXDev'05 queda destripado en un artículo que te ofrece todo el mapeado fotográfico completo.

Redacción

Francisco Álvarez
Roberto Álvarez
Óscar Centelles
David Lucena

Ilustraciones

Roberto Álvarez

Maquetación

Roberto Álvarez
Óscar Centelles

Colaboran

Frederik Boelens
Miguel Ángel Fernández
Avelino Herrera
Armando Pérez
Maarten van Strien
Jordi Tor
STAR

Contacto

http://callmsx.gabiot.com
por Avelino Herrera

EDITORIAL

Llevábamos tiempo pensando en cambiar un poco la configuración de la revista, limar asperezas y dar un look más homogéneo en lo que al diseño se refiere, ya que nos parecía un poco anárquica la distribución de artículos hasta la fecha. Así podréis observar que hemos puesto ímpetu en el orden de los artículos, divididos en secciones, para que sea más cómoda la lectura. Sólo con ver el índice ya se percibe una nueva dirección en el planteamiento de la revista.

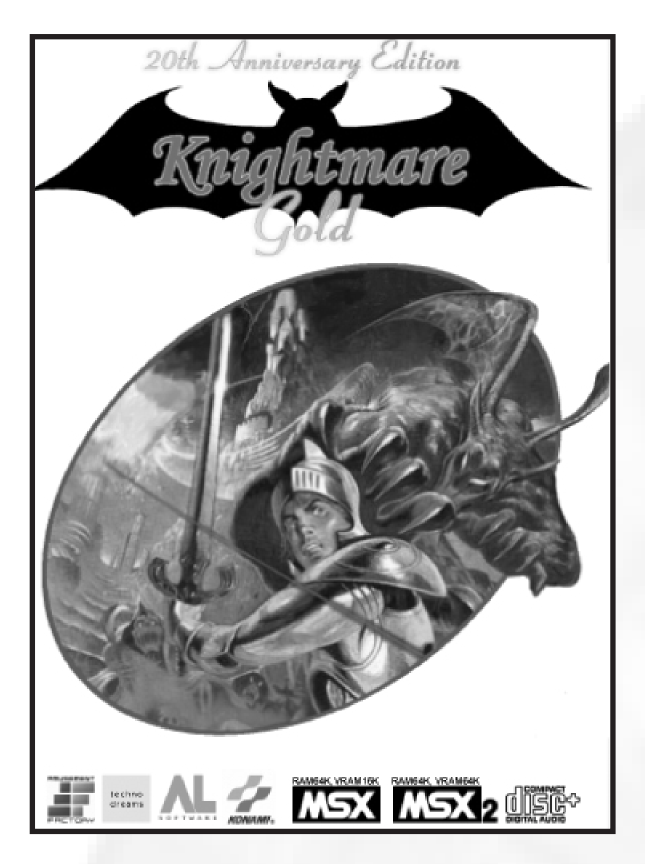
Aunque la colaboración cada vez es más escasa debemos agradecer los artículos de última hora. Muchos usuarios debido a “sus responsabilidades” ya no disponen de tiempo para afrontar un artículo con el que dar cuerpo a la revista. Tampoco pedimos eso, nos basta una opinión, un truco o el descubrir una errata en un número anterior, con lo cual seríamos felices por ver cómo los lectores se involucran en el proyecto.

Dejando de lado el mismo sermón de siempre, tenemos en este número un juego estrella, The Cure. Por méritos propios ha ganado el concurso creado por Karoshi y nos ha inspirado en la creación de la portada. Quizá la mejor noticia ha sido que la Reunión de Barcelona ha vuelto a ser numerosa, que la gente está animada con los nuevos proyectos que se avecinan en los próximos meses. Y es que se mire por donde se mire en todo podemos apreciar la palabra calidad. Los últimos juegos de MSXDev, hay varios que presumen de ella. Las últimas demos que nos van llegando son espectaculares (Sphere, MSX Unleashed) y esperamos que las próximas creaciones en formato cartucho por parte de Matra mantengan esa calidad que siempre les ha acompañado. Tampoco podemos dejar de hablar del hard, tenemos dispositivos excelentes que nos harán abandonar pronto aquellas enormes torres de PC conectadas con el Ide o las arcaicas ZIP del MegaScsi. Las nuevas tarjetas CF o MMC son asombrosas, nos reducen al mínimo espacio nuestro MSX, con lo que muchos usuarios volverán a sacarlo del armario, pese a que tengamos excelentes emuladores en estos días que corren. Y estamos ansiosos de que el proyecto Dumas vea pronto la luz, pues parece un proyecto muy interesante. Ya sólo faltaría disponer de un “nuevo” ordenador para jubilar las vetustas piezas de nuestros queridos micros ... pero esa es otra historia.

El equipo de Call MSX.

Knightmare Gold

Después de veinte años del original Knightmare de Konami, el grupo Amusement Factory Soft está cerca de finalizar un remake con un nuevo look visual.



En el anterior número comentábamos que había dos proyectos comenzados de secuelas de clásicos de Konami: Knightmare Gold y Space Manbow 2.

Knightmare Gold es el resultado de la combinación de Knightmare CD (creado en 2003 con músicas versionadas del original) más una buena parte reprogramada para su adaptación a MSX2 o superior.

Entonces, ¿qué nos aportará este remake?. Todo dependerá de la máquina y accesorios que tengáis. Si tienes un MSX1 quizá puedas jugar al juego tal y como se concibió originalmente. La novedad reside a partir de un MSX2. Y digo a partir porque con un simple MSX2 no disfrutaremos de todas las ventajas del juego. Tal y como está programado ahora mismo si queremos jugar en modo suave de scroll necesitaremos un ordenador con almenos 7 Mhz. Esto es un aspecto a pulir por los propios programadores, sinó sólo disfrutaríamos de esta opción prácticamente con un Turbo R.

También como novedad los colores de la paleta y los sprites sufrirán una mejora notable. Los colores serán más matizados, y darán un nuevo aire cromático al juego. Al marcador tradicional habrá que añadirle un par de opciones más. Y quizá lo más novedoso será la posibilidad de escuchar el juego vía CD. Tendremos en un Cd las pistas del juego creadas en formato Audio. Así que si todavía tenemos guardado nuestro antiguo Ata-Ide con la aparatosa torre donde teníamos la unidad de CD-Rom, habrá que ir sacándolo de nuevo, a no ser que se haga un driver expreso para una unidad externa vía Usb cuando dispongamos de Dumas. Aún así, de momento también podemos jugar en Psg. Aunque los programadores desearían pronto hacer las músicas para el chip SCC. Esperemos que sea cierto, pues qué buena fue la versión propia de Konami en SCC en su Game Collection.

Podéis informaros más ampliamente en la página oficial de Knightmare Gold y seguir sus progresos, así como bajaros el manual en formato PDF. Parece ser que ha habido mal rollo debido a que algún gracioso ha hecho circular alguna beta del juego sin el consentimiento expreso de los programadores. Se debería evitar estos actos que ponen en peligro la continuidad de proyectos serios que tanta falta nos hacen y respetar el digno trabajo de gente que dedica parte de su tiempo para ofrecernos productos que tanto anhelamos.

http://www.caetano.eng.br/MSXPage/KMG/







Emuladores MSX para portátiles

Todos conocemos los emuladores de MSX para Windows o Linux que son los más utilizados, pero desde no hace mucho las portátiles actuales disponen también de emuladores donde poder ejecutar nuestro MSX. Veamos cuáles son las últimas versiones aparecidas hasta el momento.

Recientemente el mundo de la emulación de MSX para consolas portátiles ha vivido una revolución. Ya no es necesario poseer un pocketPC o un móvil para disfrutar decentemente de nuestros MSX en cualquier parte.

Así, en estos últimos tiempos hemos podido disfrutar de emuladores para las portátiles de Nintendo. Si el MSX posee el famoso GEM, emulador de la original GB, son ahora sus hermanas mayores, GB Advance y Nintendo DS, las que emulan al "estándar".



Los poseedores de una NintendoDS con todo el "equipo necesario" para la misma podrán portabilizar sus MSX gracias al archiconocido fMSX, el fMSXDS, que a día de hoy va por la versión 0.06. El creador de este emulador no es otro que Nyagosu, programador japonés de diversos emuladores para portátiles. La última versión del mismo la podéis descargar desde su página web:

http://www.imasy.or.jp/~ngs/emu/



Si bien la resolución de la NintendoDS la hace muy adecuada por la resolución de su pantalla (256x192) para la emulación del MSX1, la del nuevo emulador de GB Advance requiere de un rescalado para "emularla". Ésta es de 240x160. De todas formas, el emulador GBA, MSX Advance, no se resiente demasiado por este hecho. Aunque sólo emula MSX1 y no todos los juegos funcionan, los que lo hacen son jugables. Como noticia de última hora, parece ser que la emulación del SCC va por buen camino. La versión actual es la primera release, ahora mismo la 0.1. Podéis descargarlo desde la web de su autor:

http://hem.passagen.se/flubba/gba.html

Moai-Tech nº3

Los colegas de Moai-Tech ya tienen colgado en internet el tercer número de su webzine. Se han dado prisa para tenerlo a punto antes de la reunión de MadriSX prevista para marzo.

Como en cada número cambian el fondo y la portada. Esta vez el protagonista es el personaje principal del juego Saimazoom, una adaptación de Karoshi de un juego antiguo de Spectrum.

Entre las secciones habituales destacan por encima de otras la entrevista al grupo Matra, que responderán a sus preguntas con un sentido del humor entre irónico y soberbio. Eso sí, veremos como Star es uno de los personajes con la mente más despierta del mundo MSX.

Y nuestro amigo Julio en su sección "Curiosidades MSX" nos busca la relación entre guiones de juegos y tradición judeo-masónica, desvelándonos "misterios que llevaban guardados miles de años". Cualquier día se lo llevan los hombres de negro.

http://www.moai-tech.com

INK : EXXON SURFING

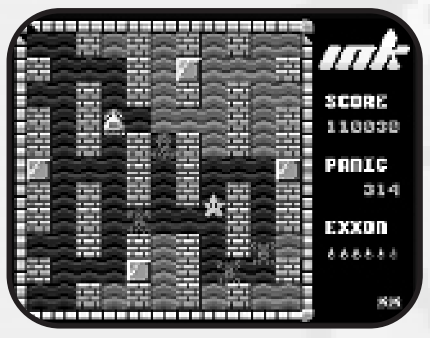
Después de muchos años por fin disponemos de un nuevo juego en cartucho para nuestro sistema. Se os hará familiar porque este juego ya tuvo una versión en casete y para la primera generación de MSX hará medio año, presentado por Matra en la 28a Reunión de Barcelona. Veamos qué novedades aporta.

De la mano del desconocido grupo The Ink Team llega una nueva versión del famoso "Ink". Y decimos famoso porque es el primer juego en años que se creaba y producía en cinta para MSX de primera generación.

Como novedad, para su versión en cartucho de este 2006, incluye una versión MSX2. Así pues y como los antiguos Konami, este Ink: Exxon Surfing detectará al arrancar la versión de MSX en la que se intenta ejecutar el juego. Si es MSX1 obtendremos la versión MSX1 que ya pudimos disfrutar en casete. Si es MSX2 disfrutaremos de una nueva versión que aprovecha las mejoras de los MSX de segunda generación.

Se espera que este Ink: Exxon Surfing esté disponible para el próximo encuentro de MSX en Barcelona. Como promoción, hasta el pasado 16 de abril podía reservarse el mismo, obteniendo de regalo un CD con la Banda Sonora del mismo. A 19.99 euros, un precio suponemos que muy ajustado, podremos tener esta joya en nuestras manos. ¿Estaremos preparados para la nueva hornada de juegos en cartucho que pueden llegar?





Nueva versión MSX2

Se abre el concurso MSX DEV'06

Tras el éxito cosechado con la última convocatoria ya se esperaba una nueva edición de este estupendo concurso. Desde el día 17 de abril se abre el período de inscripción al evento. Se dispone hasta fin de año para presentar los proyectos acabados. El veredicto lo tendremos a finales del mes de enero de 2007.

Como gran novedad decir que ya no es Karoshi el organizador de dicho evento sino un nuevo grupo de usuarios bajo el nombre de "The MSX DEV Team" (Jon Cortázar, Benoît Delvaux y Sjoerd Lammertsma).

Todos los proyectos deben mandarse en formato .Rom y como máximo 128k. Todos los juegos deben ser compatibles con la primera generación de MSX1 con un mínimo de 16k de Ram y 16k de Vram. Y por supuesto deben funcionar con el resto de ordenadores MSX, incluso con el Turbo R.

Como novedad está permitido usar cualquier hardware extra como pueda ser el SCC, el FM Pac, Music Module, Moonsound, GFX9000 ... Parece que la incorporación de este hardware será tomado en cuenta por el jurado como un aspecto positivo.

Aunque se valorará sobretodo la originalidad en los juegos creados también serán admitidos versiones y remakes.

De momento hay un único regalo para el ganador, un fabuloso MSX Turbo R FSA1GT. Un estupendo regalo por el que vale la pena presentarse al concurso.

Tiempo al tiempo, esperamos llevarnos una grata sorpresa afinales de año y disfrutar de nuevas creaciones tan buenas o mejores que las del ya pasado MSX DEV '05.

http://msxdev.msxblue.com/

Game Reader de Sunrise

La actualización de la web de Sunrise pone de manifiesto el interés por la venta de su famoso USB Game Reader, con las mismas características que la versión de ASCII

Los chicos de Sunrise han conseguido dar oficialidad a su nuevo proyecto al obtener el acuerdo definitivo con Bazix. Así pues, su venta es una realidad sin problemas legales.

El dispositivo viene con las mismas características que el MSX Game Reader oficial, así como con sus detalles "polémicos". Por lo tanto, no será posible usar cartuchos que necesiten de +12V ni que sean sólo de I/O (es decir, sin memoria). De todas formas, tras lo hablado con los miembros de Sunrise en la pasada reunión de Bussum, están preparados para hacer las modificaciones hardware pertinentes para conseguir que dichos cartuchos funcionen. Como novedad, la versión del MSX PLAYer incluido es una versión traducida al inglés. Esto hace mejorar el package original, en el cual sólo podíamos obtener dicho emulador en perfecto japonés.

http://www.msx.ch/sunformsx



Vuelve Pac-Man

Otra versión del clásico inmortal de Namco está siendo desarrollada para MSX2 por Eduardo Mello de Opcode Games. Sin embargo, no se tratará de una versión más, sinó una que promete ser la más fiel a la recreativa de 1980.

Además, al tratarse también de un recopilatorio, se incluirá el Pacman original (incluyendo la versión japonesa, Puckman) y el adictivo Ms Pacman y se incluirán todos las animaciones, gráficos, sonidos y melodías de éstos.

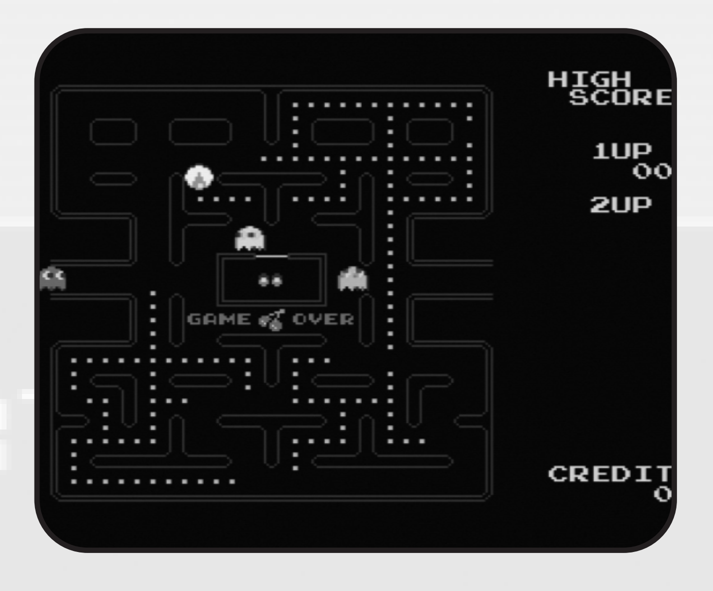
Un detalle interesante es que el programador hace uso del S.C.C. para mejorar la calidad del sonido.

¡Pero ahí no acaba todo! Para acabar de poner la guinda al pastel, el juego será editado en cartucho de 1Mb y será presentado en su estuche, con su manual y todo; "packagin" gracias a Dale Crum y Jess Ragan.

Si quereis haceros una idea de cómo será el resultado final podéis bajaros la demo que hay disponible en la página de Opcode Games. La demo incluye únicamente la versión del Pac-Man original y solamente se puede jugar con joystick.

De todos modos podéis echar un vistazo a la versión Ms PacMan en la página web de los creadores.

http://www.opcodegames.com/games/pacman/PAC_MSX.rom



28ª Reunión de Usuarios de MSX de Barcelona

Después de la incertidumbre sobre el número de asistentes en la anterior reunión aquí comentada, podemos decir que sí, que efectivamente fue un error el proponer una reunión en apenas dos semanas de antelación. Los que asistimos el día 3 de diciembre perdimos la angustia al ver como la sala se llenaba como en los viejos tiempos (Según las entradas vendidas hasta mediodía 74 asistentes, aunque por la tarde aparecieron algunos más). Parece que la oferta, las noticias de nuevo hardware y la llegada de usuarios holandeses consiguieron atraer a los usuarios hasta Sants. La verdad es que daba gusto contar con caras míticas de la escena española que hicieron un gran esfuerzo por venir a Barcelona, y lo más sorprendente, ver caras nuevas, lo que hace pensar que se recuperan usuarios descolgados. Hay que decir que echamos de menos nuevamente a los usuarios de las islas Baleares, el grupo Taburoto o los bilbaínos Lehenak. Pero bueno, no se puede pedir todo.



Panorámica de la sala

Como el anterior evento, se celebró en la sala "El Altell", En la última planta de un edficio anexo al edificio de las Cocheras de Sants donde antiguamente se realizaban las primeras reuniones. Se pagaba una entrada de 3 euros nada más subir el último peldaño de las escaleras, dentro del stand de segunda mano que como en cada ocasión lo presidía Jordi Tor. Con el precio de la entrada se regalaba un Cd de música (los 80 primeros) que Zanac, miembro de Call MSX, recopiló con temas arreglados de los mejores compositores de Msx en Moonsound. Y es que este stand estuvo en constante evolución de productos sobre la mesa. Habían ordenadores, Music Modules, un track ball, algún teclado para midi y sobre todo muchos juegos como de costumbre.

Immediatamente después estaba nuestro stand, donde como novedad vendíamos nuestro quinto número, y como siempre lo que quedaba de números atrasados, entre ellos la edición especial para Bussum en inglés, como oferta para rezagados o nuevos usuarios que se incorporan. También en nuestro stand se vendía la última recopilación musical de Jordi Tor, MSX Perfect Covers III, y sus anteriores entregas.

A nuestro lado teníamos a Manuel Pazos que se trajo un



Stand de Segunda Mano



Blox, el nuevo proyecto del Team Bomba

prototipo de MSX One Chip insertado en la carcasa de una Playstation 2. La verdad es que es un cacharro espectacular y que mantuvo la expectación entre los curiosos usuarios. Se podía comprobar in situ cómo funcionaban juegos de la talla de Solid Snake o SD-Snatcher. Pero lo más curioso fue cómo también era capaz de leer cintas de audio. Quizá es una buena noticia para los usuarios más escépticos que no apostaron en un primer momento por dicho proyecto, y que si hay una segunda convocatoria de pedidos se animen y por fin tengamos nuestro antiguo MSX, el mismo, en un "cuerpo nuevo".



Stand de Call MSX

En la misma mesa junto a Manuel Pazos estaban los chicos del Team Bomba. Vinieron en total cuatro miembros y nos enseñaron su nuevo proyecto, Blox, exclusivamente para Graphics 9000. Sólo pudieron mostrar una demo con logotipos de alta resolución y un editor gráfico propio para la creación pantallas. Lástima que se dejaron en Holanda el motor del juego y no pudimos ver algo más. Está inspirado en un juego de Flash del mismo nombre en el que debemos mover unos iconos determinados para podernos pasar la pantalla correctamente. Son ese tipo de juegos sencillos pero que van creando adicción a medida que las pantallas se van volviendo más complejas para resolver. La verdad es que fueron unos chicos muy simpáticos y agradables y estuvieron en todo momento haciendo relaciones públicas con todos los stands. También trabajaban en el portátil desde el emulador OpenMSX y estuvieron intercambiando conocimientos con Armando Pérez y con los miembros de Karoshi. La verdad es que se agradece que gente de fuera venga a visitarnos, aparte de su exquisito comportamiento.



El MSX One Chip en pleno funcionamiento



El stand de CDSG

A continuación estaba el stand de Carlos de Santana García, que pese a no presentar nada oficial estaba programando en su Turbo-R una utilidad Irc para la Obsonet, en consecuencia, también será válida para los futuros poseedores de la tarjeta Dumas.

Avanzando nos encontrábamos con el stand de Paxanga que aparte de vender sus antiguos productos (Yupipati) nos



Los creadores de Parachuteless Joe

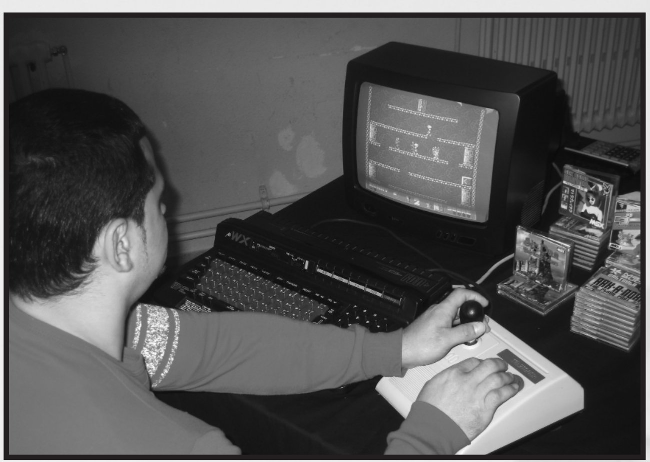
presentaban su nuevo proyecto que se presentaba en el concurso de MSX Dev'05, Parachuteless Joe, al cual podíamos jugar para probarlo. Ya fue comentado en la revista anterior en la sección del concurso. No sabemos si cuando acabe el concurso tendrá intención de venderlo empaquetado o por razones de las bases del concurso que desconozco no lo hará, ya veremos.

En el siguiente stand encontramos al grupo Matra junto a Kralizec. En estos momentos son los grupos que lideran la creación de software para nuestro sistema en cuanto a proyectos ambiciosos se refiere. Vendían cosas diversas,



Productos de Matra y Kralizec

como camisetas con diseños propios cambiando el concepto original en los que se basan los logos. Vendían un par de cassettes para Msx1, uno de los cuales era una adaptación de un juego original, Bloody Paws, programado en Spectrum y



La novedad de Kralizec, Dahku

que lamentablemente le fue "robado" por la empresa GLL a su creador. Era una manera de hacer justicia 15 años después de la indebida apropiación. Vendían juegos antiguos de la propia Matra y de Kralizec. Éste último presentó su novedoso Dahku del cual ya hablamos en el pasado número. Pero quizá lo más importante fue la charla que dió Star sobre su nuevo cartucho sonoro y de la que pudimos ver una pequeña demo en



El stand de Karoshi

funcionamiento. Parece que la cosa promete y esperamos tener pronto en nuestras manos un cartucho nuevo con este chip de sonido revolucionario.



¡Tiro al pato!

28ª RU Barna

Otros habituales son los chicos de Karoshi, quienes nos ofrecían en su stand los últimos proyectos en los que están immersos (el finalizado Saimazoom y los inacabados Columns y Griel's Quest For The Sangraal), todos ellos se presentan al

El stand de Moai-Tech

concurso de la MSX Dev'05. Fueron cargando algunos de los programas que participaban en dicho concurso y por los rumores que se cocían en la sala, el juego Caverns of Titan despertó muchos elogios, por su preciosismo gráfico y porque recordaba enormemente a clásicos de la talla de Manic Miner o Blagger de Alligata.

El último stand era el de Moai-Tech. Recordar que publicaron su segundo número del webzine y traían carcasas de cartuchos creadas por Ángel Alonso. Además vendían el Cd musical que ya presentaron en la pasada reunión.

Pues la reunión pareció estar tan entretenida que incluso se olvidó que había un concurso preparado. Llegaron las 5 de la tarde en un santiamén y tocó recoger el chiringuito. Como colofón final se hizo una cena de hermandad, batiendo récord de asistencia, que ya viene siendo habitual después de cada evento barcelonés.

Y eso es todo lo que se coció aquel 3 de diciembre de 2005. Esperamos que las sucesivas reuniones puedan mantener un ambiente similar en cuanto a número alto de visitantes y por supuesto a una regular presentación de novedades.



Holandeses en Barcelona

La estrella de la Reunión

Yo soy el ombligo del mundo

MSX DEV'05
-continuación-

En el número anterior comentamos este concurso con los proyectos que hasta la fecha parecían acabados o en la última parte de su desarrollo. Pero, la verdad es que hubo sorpresas de última hora las cuales no pudimos comentar por aparecer posteriormente. ¡Y qué sorpresas! La verdad es que aparecieron juegos realmente prometedores que han dejado el listón muy alto para próximas convocatorias. Vamos pues a ver qué juegos son estos que no vimos la otra vez.

En los días previos al cierre del concurso se presentaron algunos juegos más, entre los cuales estaba el que sería el elegido como vencedor. Así que hemos pensado en la redacción en acabar de comentar los juegos que no pudimos hacerlo en su momento y que también merecen unas líneas en esta revista.

Universe : Unknown

Infinite es el grupo creador de este shoot'em up de concepción nemesiana que nos sorprende con un scroll horizontal con dirección de derecha a izquierda, vamos, con la dirección invertida a lo que Konami nos tenía acostumbrados en su saga Nemesis. Parece que Infinite tenía previsto acabar el juego fuera de concurso, hacerlo más elaborado, ya que hay algunos errores con las colisiones que por falta de tiempo no deben haber podido solventar. Tiene una intro larga con abundante texto y gráficos simples y un poco repetidos.

El juego en sí es poco original, ya que se inspira en exceso en el Gradius, aunque también tiene toques que recuerdan al R-Type. Cambia el modo de selección de armamento pero los gráficos, que son excelentes, están demasiado en sintonía con el clásico de Konami. Se asemeja mucho la primera pantalla a una fase de la beta del Gradius 2. Pero bueno, no debemos quejarnos tanto porque lo importante es que es muy jugable, mantiene una calidad gráfica asombrosa, qué buenas son esas plataformas con esas hermosas cabezas femeninas, y no le vamos a echar ascos porque sea poco original.





La música está pero que muy bien, la melodía tiene mucha calidad con unos acompañamientos más que logrados, es una gozada ver cómo el maestro Wolf pasa de hacer virguerías en Moonsound a hacer temas geniales en Psg con la misma facilidad. Los efectos de sonido también están a a la altura.

La verdad es que teníamos muchas ganas de volver a tener un juego de estas características. Es que un buen shooter es sinónimo de mucho vicio.

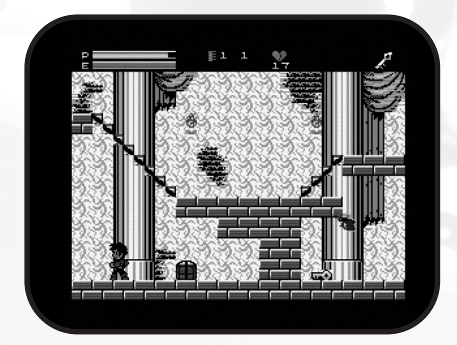
The Cure



Lo cierto es que los que hemos podido apreciar este juego antes del veredicto ya imaginábamos que estábamos ante el vencedor del concurso. Este juego es increible. Parece mentira que un juego, también calcado de un clásico de Konami, de sólo 48k y en Msx1 pueda compararse codo con codo con el original Vampire Killer. Estamos ante un juego realmente al estilo consolero. Esto quiere decir que tenemos ante nosotros un juego que se mueve el personaje como pocas veces lo he visto antes en un MSX. Es fascinante, es para sentirse orgulloso y mostrar a los colegas de otros sistemas lo que se puede hacer en MSX1 con tan poca memoria.

El culpable de este engendro es el grupo XLS2 Entertainment, quien ya haya visitado su página verá que esconde algún proyecto muy interesante para el futuro. De buen principio nos sorprende con su logo en el que aparecen animados unos murciélagos que se mueven con una facilidad

pasmosa. Si lo dejamos pasar veremos una intro sencilla con gráficos muy pixelados y donde nos explica de qué va el juego.



Una vez pulsamos para jugar, lo primero que nos llama la atención es el tamaño del personaje, si lo comparamos con el Belmont estilizado de Msx2, y cómo se mueve de suave uno y otro. Después de este primer tacto descubrimos que los gráficos son muy buenos y hace un uso espectacular de la pobre paleta de nuestro ordenador. El truco se debe a un inteligente diseño y maestría en la pixelación a modo de trama. Podemos apreciar en la montaña en forma de calavera del segundo plano cómo sobre un fondo rojo se dibuja con negro a líneas intermitentes y crea una sensación de mezcla de color marrón y da la sensación de quedar alejado. Lo mismo ocurre con su reflejo en el pantano. Parece sencillo pero para llegar a esa síntesis hay que estrujarse antes el coco. La única pega son los sprites del Msx1 que no disponen de transparente y a veces se ve el cuadradote negro donde no hay dibujo. Pero no se pueden pedir peras al olmo.

Luego nos ha gustado mucho que cuando el personaje pisa un arco, éste se hunde y debes darte prisa en saltar al siguiente. La parte incial nos recuerda mucho al primer Castlevania de la Super Nintendo, y la verdad es que está bien que no sea todo una burda copia del original Vampire Killer. Dentro del castillo ya sí que la inspiración es de casi el 100 por cien, por lo menos en la primera fase. Debemos ir cogiendo las llaves para luego luchar con el enemigo de turno. La segunda fase tiene un colorido magnífico y nos recuerda a las paredes del mítico Castillo de Cagliostro. La música es bastante sencilla en Psg, quizá lo más flojo del programa, en cambio los efectos son correctos.



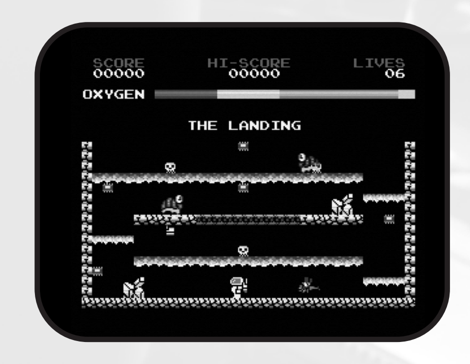
En definitiva, un juego que si ha merecido ganar es porque supera con creces a todos sus rivales. Está en otro nivel, el profesional. La verdad es que es satisfactorio poder comprobar que no sólo Kralizec nos puede aportar juegos de altísimo nivel, sino que comienzan a salir títulos admirables en otros grupos. Esto quiere decir que los programadores amateurs de Msx por fin están alcanzando un nivel digno, lo que todavía les dará ánimos para que se atrevan con nuevos retos y nos sorprendan con grandes títulos para tiempos futuros.

Caverns of Titan

José Luis Tur Santolaria más algún colaborador nos traen este juego magnífico de plataformas. La presentación al más puro estilo Namco y con un scroll de texto que nos explica que Willy, el protagonista, debe ir recuperando chips para conseguir el oxígeno que va perdiendo poco a poco.



El juego nos recuerda mucho al Manic Miner, Jet Set Willy (de ahí el nombre del protagonista) o Blagger en cuanto al concepto. La verdad es que gráficamente es bonito y colorista, están los gráficos muy definidos con animaciones de los personajes simples pero correctas, parece una Rom clásica profesional. Hemos de decir que si lo ejecutas en Msx2 la paleta sufre una mejoría cromática, vamos que aprovecha las posibilidades de la máquina superior. El amigo José Luis ha conseguido dar esa imagen colorista que iba buscando. Willy se mueve perfectamente aunque nos hubiera gustado un pelín más de velocidad, pero bueno todo queda bien ajustado. Debemos salvar a los enemigos saltando e ir recuperando los chips. Las pantallas se van complicando, a veces la dificultad es excesiva, y hay que estudiar el camino correcto para poder cumplir con éxito la misión. La música suena muy bien, son buenos temas que dan buena ambientación al juego y los efectos también dan la talla.



Saimazoom

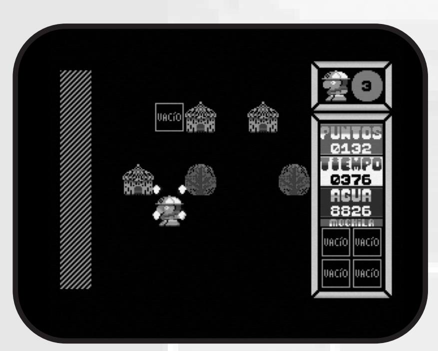
Karoshi Corp. nos presenta una versión de un juego antiguo de la casa Dinamic (1984) para los ordenadores Spectrum. Este juego era la primera parte de una trilogía en que su última parte, Abu Simbel Profanation, fue la que cosechó más fama.

Pues viendo que se trata de una fiel conversión diremos que el aspecto gráfico pese a su fidelidad nos gusta más en esta nueva versión que en la original. Los gráficos aún siendo sencillos están más trabajados por Karoshi. Quizá tenga algo que ver la diferencia gráfica de cada máquina.



El explorador debe encontrar 4 sacos de café por un vasto mundo de 100 pantallas y por el complejo camino irá encontrado armas y objetos que le facilitarán la labor.

Estamos ante un buen juego, distinto de lo visto en Msx, con música correcta y entretenido.



The Mansion

Yermani Soft nos presenta en esta ocasión una aventura gráfica de las que pocas hemos visto en nuestro sistema en cuanto a nuestro idioma se refiere. Recordemos que AVG's nipones habían muchos pero nos resultaban injugables por la barrera idiomática.



Lo primero que debemos conseguir es comprender qué significa cada icono para poder desenvolvernos.

Los gráficos parecen abstractos debido a la poca definición, pero nos sitúan bien en la escena. La música es prácticamente nula.



Cheating wives

Un original juego creado por Crappysoft. Nos puede recordar en cierto modo al genial Colt 36 pero es bastante distinto en cuanto a temática. Ya que aquí debemos encontrar a nuestra señora en alguna ventana del edificio. Tenemos 5 intentos para encontrarla. En las ventanas aparecen chismosas que con sus miradas nos indican dónde puede estar nuestra querida. Pero a veces mienten, así que no será la cosa tan fácil.



Los gráficos son muy sencillos pero divertidos. La música tampoco es muy compleja que digamos, cumple su cometido pero sin alardes. Quizás la pega sería la respuesta de los controles, un poco tosca. Pero aquí lo gana todo la originalidad.

Manicomio

Un original juego conversacional de Crappysoft al más puro estilo Perry Mason pero esta vez la historia se centra en un manicomio, ya que el protagonista ha sido encerrado después de matar a su mujer a hachazos.





Para poder avanzar en el juego deberemos usar el teclado e ir escribiendo verbos típicos como examinar, coger, dar, comer, abrir ... Quizá lo más destacado es que el manicomio tiene su rutina horaria y las cosas que hagas dependerán en gran medida de la hora del día a la que las realices.

Como punto negativo decir que carece de sonido y que sólo está disponible en español. Lo sentimos esta vez por los usuarios extranjeros.

Crazy Buggy

Crappysoft también es el autor de este juego, su primer programa en ensamblador. La verdad es que es un intento de imitar al City Connection o al Choro-Q, pero se queda en eso. No hay scroll, sólo debemos llegar hasta la cima de las plataformas. Pasamos de una a otra de una manera un tanto cutre. Debemos ir salvando los obstáculos saltando con la barra espaciadora.





La verdad es que poco más podemos decir de este programa. La música sí es divertida y encaja bastante bien, pero sin ser muy elaborada.

Namake's Bridgedrome

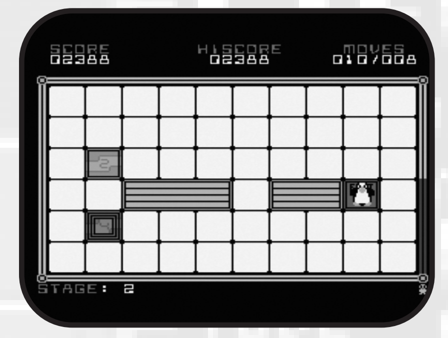
Un juego de puzzle creado por Buresto Faiya y que está inspirado en un concepto de juego matemático llamado "The Kung Fu Packing Crate Maze" disponible en una página de internet.

Debemos mover a nuestro pingüino moviendo unos bloques con un número diferente en cada uno hasta llegar a la salida para pasar de pantalla. El objetivo es conseguir la almohada del pingüino que ha sido robada.

El juego posee unos gráficos decentes para el tipo de juego que es.

La música es una melodía bastante famosilla del mundo del cine, aunque es bastante sencilla. El juego es complicado, tienes que pensar muy bien los movimientos que piensas realizar, y además hay una barra de tiempo para acabar de complicarlo. Lo bueno es que puedes usar passwords según las fases que te hayas pasado.



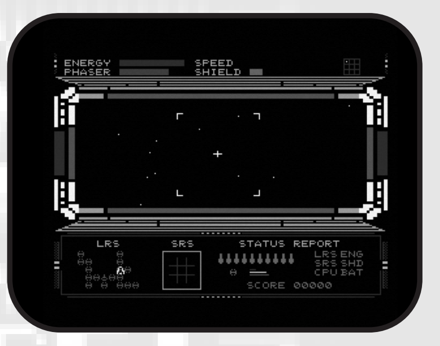


En definitiva, un juego curioso que te hará pensar y te distraerá un rato.



Space Crussader

Aquí tenemos un simulador espacial creado por Yermani Soft. Este tipo de juegos son más fáciles de encontrar en máquinas como el Pc, aunque en Msx también había algún que otro título con esta manera de jugar, como el Flight Simulator with Torpedo Attack.



Hemos de reconocer que este tipo de juegos se nos hace "raro" y casi no sabemos con qué varemo comentarlo.

Si leemos las instrucciones que lo acompañan veremos que el control es más complejo de lo habitual en un videojuego, ya que usa muchas teclas para realizar diversas funciones, lo cual hará desistir a los considerados jugones fáciles, los que sólo quieren una cruceta de dirección y un par de botones de fuego.

La pantalla parece bastante correcta. Un gran visor del interior de nuestra nave donde visualizaremos el espacio exterior, simulando una profundidad real. En la parte superior tenemos las típicas barras de energía, velocidad y escudo, y en la parte inferior el status del jugador y un pequeño rádar donde iremos viendo por dónde se acercan los enemigos.

Resumiendo, Msx-Dev '05 ha sido un éxito sin precedentes. La gente se ha animado a hacer programas y eso es lo más importante. Esperemos que como dice el dicho "el que prueba repite" y no sea la última vez para gente que ésta ha sido su primera experiencia en la creación de un videojuego. Y para los que ya están en otro nivel sea una buena oportunidad para ir superándose en cada convocatoria.

¡Hasta la próxima!

Maarten Van Strien

- WOLF -

Si la pasada entrega conseguimos traeros la traducción de la entrevista que la MSX Magazine hizo a Hideo Kojima, un valuarte de la época comercial del MSX, en este número os presentamos la entrevista que hemos realizado a un famoso MSXero de la época post-comercial. Maarten van Strien, más conocido como Wolf, es quizás el compositor más prolífico de la escena del MSX en los últimos 15 años. PSG, FMPAC, FM + Music Module, Moonsound... muchos chips sonoros han pasado por su buen hacer como músico.

[Los primeros años]

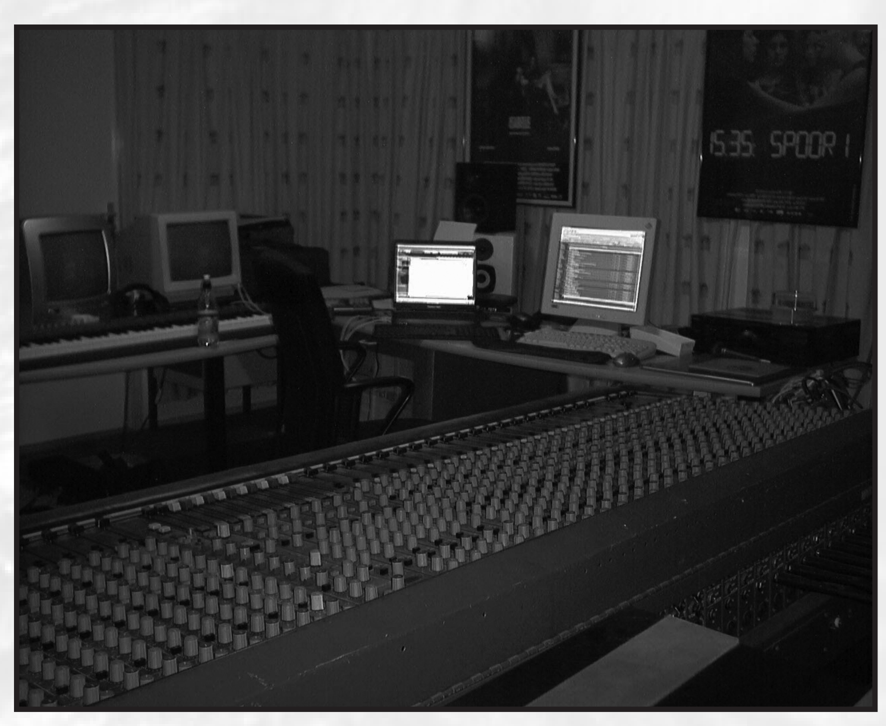
-¿A qué edad comenzaste con el MSX y cuál fue tu primer modelo?

Mi primer MSX fue el Toshiba "el tanque británico" HX-10 y creo que yo tenía unos 10 años, año arriba año abajo. Principalmente estuve tomando los primeros pasos con el BASIC con la ayuda de libros, nada espectacular. Imagino que era bastante normal por entonces aprender a programar tu ordenador. En el colegio había MSX2 Philips VG8235 y el Designer de A. Koene era el programa que utilizábamos. Naturalmente no funcionaba en mi MSX1 por lo que pronto conseguí un Philips 8245 con el que programé tonterías y jugué. De alguna forma veía el arte por ordenador bastante interesante porque era más fácil trabajar con el Designer o con el BASIC si lo comparaba con el lápiz y el papel. Sería irreal pensar que llegaría a ser un compositor porque a decir verdad no me encaminaba en esa dirección. La primera vez que recuerdo que compuse algo fue en el colegio, para una pequeña obra de títeres. Toqué esa melodía en un viejo piano que realmente estaba desafinado. Sin embargo esto no se percibió y hubo varios grupos que hicieron obras con títeres. Creo que debe haber sido la primera vez que ganaba una competición en la que participaba. :) (¡Hola Snout!) La obra tenía dos piezas de música, un tema positivo medieval y otro más oscuro que estaba inspirado en la música del espacio del Topple Zip. Al año siguiente esto "salvó mi culo" ya que no necesitaría cantar cada lunes por la mañana con el resto de la clase. De esta forma tuve que arreglar todas las canciones para piano con la colaboración de mi profesor de piano. Mi 8245 está ahora en la habitación de al lado, disfrutando su jubilación. Mi configuración actual es un 8250 con 2 disqueteras, V9958, 256KB internas de RAM, lector de CF, Moonsound con 640KB y un GFX9000 conectado a otra TV. El FMPac, el Music Module y el SCC están con el 8245.

¿Qué estudios musicales seguiste?

Como crecí sobre una tienda de música que se mantuvo hasta 2002 podía más o menos tocar antes del comienzo de las clases de piano. También manejé órganos enormes, desde humildes Solinas a enormes Eminents. De los 7 hasta los 17 tomé lecciones de piano orientadas a la música clásica, con deslices diarios con el boogie 'n blues, romántico moderno e incluso tocando temas del Synthesizer Greatest como lecciones. ¡Puedo decirte que Axel F. está chupada! Entonces no me veía como un compositor ya que tenía otros hobbies no relacionados: dibujante sobre papel, astronomía... Incluso el MSX no estaba prácticamente relacionado. Todo esto cambió cuando comencé a leer anuncios en la MCM (Dutch MSX Computer Magazine) donde alguien en mi pueblo distribuía "discos copiados". ¡Para mí eso eran obviamente muy buenas noticias! :) A través de ese contacto también conocí el software de dominio público como los famosos Genic Picturedisks. El primero fue el número 7, con la famosa promo de Royal Art sin música. Por entonces experimentaba con un FMPAC (el coreano) en BASIC y desde ese momento me adentré más en la composición. Coincidió con que RMF buscaban desesperadamente un compositor y les envié una carta. Yo tenía unos 13 años y unos 6 meses de experiencia con el FMPAC en BASIC. Me contrataron, pero tuve que utilizar el FAC Soundtracker combinando FMPAC y el Music Module. Por suerte un cliente de nuestra tienda de música tenía un Music Module con el que no hacía nada y pude quedarme con él. Así pues, ¡la falta de Music Module había desaparecido! Por esas fechas el FST era un completo desconocido para mí y me costó tiempo acostumbrarme hasta que logré dominarlo hasta el punto en que componía sin monitor, ya que estuvo roto por un corto espacio de tiempo. :) Un día hice una demo del RMF music disk #1 y se la enseñé a mi

profesor de piano. Se llamaba "Approaching the Force". Él me comentó que podía estudiar composición en un conservatorio aunque hasta entonces yo creía que era para intérpretes y directores. Así que el hecho de que él lo mencionara fue el comienzo de lo que podría llamar "composición en serio". Después de visitar un conservatorio en Amsterdam decidí que era demasiado artístico... por llamarlo de alguna manera "algo que no nunca vende". Entonces fui a La Haya porque me llamaron tras recibir mi demo y decidimos que su instituto estaba lejos de lo que yo hacía (usar la tecnología). Años después encontré un nuevo instituto en Hilversum que estaba hecho a medida para mí. Básicamente era todo lo que yo hacía por lo que acabé allí desde 1997 al 2001. ¡Esos años son la mejor excusa del porqué yo estuve tan parado respecto al MSX!



-¿Empezaste creando melodías para MSX en PSG? ¿Fueron "programadas" en BASIC?

Depende, obviamente mis primeras "cosas" fueron hechas en BASIC, pero llamarlas melodías sería demasiado honor, creo. Las cosas mejoraron con el FMPAC, pero nada cercano a una "estructura" decente. Las estructuras reales vinieron con FST, quizá debido a mi educación clásica y la analogía entre los trackers y la notación clásica.

-¿Preferías el sonido SCC de Konami o las melodías FM de compañías como Micro-cabin durante la época dorada del MSX?

Es muy complicado decidirse. Los juegos de Konami con SCC son casi todos juegos de acción. Los únicos RPGs reales serían el Snatcher y el SD Snatcher y puede que el Solid Snake, aunque no estoy seguro si se clasifica como RPG. SD Snatcher es cyberpunk mientras que las series XAK/Fray son de fantasía/medievales. El sonido de los SCC es más sintético mientras los sonidos del FMPAC se acercan mejor a instrumentos reales. Ambos SD Snatcher y la serie XAK tienen el perfecto chip de sonido para su música, así que yo no me atrevería a intercambiar los chips en esos juegos.

- ¿Fue interesante para ti trabajar con el FMPAC y el Music Module al mismo tiempo (el llamado Stereo)?

El Stereo fue la forma en que el Moonblaster fue en parte promocionado. Fue una pena que un único canal sirviera a ambos chips (realmente no te atreves a desaprovechar un canal seleccionando un único chip) no importa cómo o qué: el FM y el Music Module son dos chips diferentes, ambos merecen ser optimizados con tratos diferenciados. Sin embargo Muzax 3 probó que podías conseguir resultados decentes. En el caso ideal que se pudiera se desearía tener 18 canales independientes en vez de 9 compartidos.

-Ya que hablas de eso: ¿te sentías limitado con los trackers como el Moonblaster?

Al principio no, pero porque simplemente yo no sabía hacerlo mejor. Al final lo encontré un poco aburrido ya que mi objetivo había sido siempre conseguir recrear la música de Microcabin la cual simplemente no puede hacerse con el MB. Esa fue una de las razones por la que dejé el MSX tras 1996 junto con otras como el alza de los PCs y sus posibilidades sonoras (todos debemos mejorar) y mi ya comentado estudio del '97 al '01. Aunque generalmente esos años son bastante silenciosos en el MSX ya que que fue un poco como la tendencia.

-Al menos en España, el MIDI en el MSX no tuvo un gran éxito. ¿Qué opinas del MIDI en el MSX? ¿Lo probaste? Si es así, ¿podrías darnos tu opinión de software tipo Synthsaurus?

Vaya, no me enciendas... :) MIDI tiene un enorme problema: es únicamente un estándar de formato, no un estándar de sonido. No hay forma en la que pudieras predecir cómo las cosas sonarían en otro sistema, lo cual provoca que no tenga utilidad para gente que, como yo, tiene bastante intereses en el diseño de sonido y en la producción. El estándar General MIDI generó compatibilidad pero a parte de eso fue todavía oscuro para los diseñadores de sonido. Lo que quiero decir es:

¿Hay quien prefiere la versión MT32 de Illusion City a la versión original en FMPAC? Yo creo que no. Lo mismo sucede con el GM... no puedes sólo transformar las estrictas voces FM a samples no relacionados en GM.

El MSX se ayudaba más con un chip sonoro como el Moonsound, algo que sonaría igual en todos los chips. Naturalmente ahora tenemos el tema de la SRAM del Moondsound llevando a las mismas situaciones donde una melodía no funciona en todos sitios si no hay suficiente SRAM para cargarla. Pero eso es algo que puede subsanarse si se espande el Moonsound de 128K hasta los 640K por el precio de una película reciente en DVD.

¿Synthsaurus? Lo vi un poco por encima una vez y es completamente lejano a la velocidad de composición que tendrías con FastSoundTracker o MB. Lo que queda es un conjunto de dibujos de dinosaurios rodeado con algunos esquemas y tablas relacionados con la música. No subestimes el cambio cultural que trajo el FST. El MSX cambió de la nada a un sistema musical. Y el Synthsaurus era claramente una aplicación que no tiene su sitio en esta nueva cultura.

[La época de las Demos]

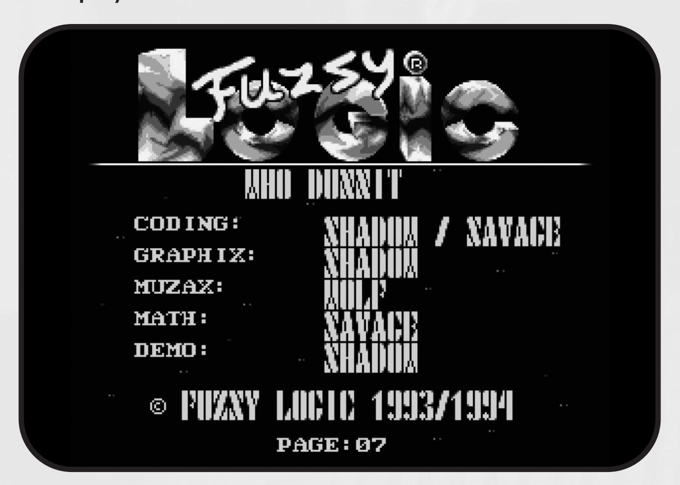
-¿Fue Royal Art de la Royal MSX Force tu primera demo? ¿Podrías explicarnos alguna anécdota interesante de entonces?

Sí, ésta fue mi primera contribución a la "scene". A parte de eso, la demo estaba ya casi terminada cuando me uní a RMF por lo que si buscas anécdotas deberás preguntar a Edwin (Edwin_ en el MRC). De todas formas algunas melodías en esa demo fueron realmente algunas de mis primeras canciones en el FST mientras otras ya estaban adelantadas, ¡por eso hay bastante diferencias entre ellas!



Megademo de Royal MSX Force (1993)

-¿Por qué te uniste más tarde a Fuzzy Logic? ¿Colaboraste en la segunda y tercera parte de Muzax y además en algún otro proyecto?



MB Muzax 2 de Fuzzy Logic (1994)

El RMF murió por los estudios de ingeniería aerospacial que Edwin ya cursaba cuando yo llegué. Eran estudios bastante complicados por lo que no tenía mucho tiempo para dedicarle al MSX. Tanto él como yo éramos básicamente los únicos miembros activos por entonces.

Por lo tanto, seguí los consejos de Bart (Cain en MRC) y llamé por teléfono a Savage quien me redirigió a Shadow. Les dije "habéis conseguido un nuevo compositor" y se quedaron bastante sorprendidos pero aceptaron. Desde ese momento comencé a coleccionar viejos temas que no habían sido hechos públicos para MB Muzax 2. Tras MB Muzax 2 Shadow y yo comenzamos a trabajar en un juego de plataformas llamado Flicky que luego se conocería como "Fubsy". Realmente nunca despegó, una pena porque los gráficos y la programación eran muy buenos. Ya se sabe, cuando está sobrio, ¡puede crear píxels muy decentes! :) Creé unos pocos temas que luego reaparicieron en Muzax 3 y como canciones de demostración para Oracle. Tras eso la escena descendió un poco y el resto es historia de nuevo. Muzax 3 fue bastante cuando salió pero marcó también el final de mis intereses en el MB Stereo y un poco en la música para el MSX en general.

-Durante la época dorada del software holandés para MSX había otros bien conocidos compositores como Hans Cnossen. ¿Qué opinas de sus canciones? ¿Crees que eran de un nivel tan alto como los músicos de los juegos comerciales de MSX?

Hans era un poco como Knightram de ANMA, algunas veces conseguía temas pegadizos, pero dios mío: el volumen y la dureza de los sonidos podrían haber sido menos acentuados. :P Es duro comparar el material de la "scene" con el de los

juegos comerciales si creo que hablamos de los juegos japoneses ahora mismo. Primero de todo, la música de la "scene" estaba directamente impuesta por nuestros programas. MB dirigió a la gente hacia un cierto estilo de música, simplemente porque la gente tendía a trabajar eficientemente, usando copias, bloques, etc. Podías escribir cada nota separadamente, por supuesto, enmascarando el hecho de que tú estás usando una herramienta sensible a los bloques y a las copias, pero ¿quién lo ha hecho alguna vez? Yo seguro que no. Los compositores japoneses probablemente usaron alguna herramienta tipo MML, quizás incluso MIDI, no lo sé.

En segundo caso me imagino que los compositores japoneses de juegos eran profesionales, compositores reales que es algo opuesto a muchos "sceners" que aprendieron por sí mismos. Esto no quiere decir que la "scene" no tuviera compositores reales, pero era todo tan fácil hacer música en el MSX que no tenías que ser un profesional de la música para conseguir algo decente de un tracker.

De todas formas y para ser honesto creo que para músicas reales de juegos se tenía que confiar en Japón, ya que la música en un juego dentro de la "scene" era un poco como tener un "musicdisk". No era realmente música dedicada 1:1 al juego tal y como la música de una película se compenetra con la película en sí, en términos de estilo. Esta relación 1:1 es algo que realmente quiero conseguir con los futuros juegos que estoy haciendo. Para Realms of Adventure (ver la web de Sunrise) el estilo musical será enorme en sonido, pero encajará siempre en el arreglo. Teniendo música 1:1 significa que el jugador se ve envuelto en el juego fácilmente y para RoA esto es muy importante: el juego será enorme, masivo, gigante y vasto. ¡Tened la seguridad que seguís su desarrollo y su venta, y estad seguros de que vuestros Moonsounds tienen 640K!

-...Y entonces llegó el Moonsound. Fue un dispositivo caro de mediados de los 90, al menos para la mayoría de los usuarios españoles. ¿Cómo crees que este cartucho contribuyó a la música en el MSX? ¿Qué podrías comentarnos sobre las partes FM y WAVE?

El Moonsound es la extensión musical más avanzada en el MSX. Desafortunadamente contaba con poco respaldo por parte de Sunrise (para ser más preciso: Remco Schrijvers) entonces. Por lo tanto, MBWAVE y MBFM fueron sólo rápidos upgrades del MB. Y ok, es algo normal para los primeros meses e incluso el primer año, pero ahora estamos esperando la salida de Meridian mientras Marcel Delorme hizo las actualizaciones de MBWAVE y MBFM en los últimos años. Ahora estamos en 2006 y más de 10 años han pasado desde el lanzamiento del Moonsound mientras esperamos un tracker decente. ¡Esto es malísimo! Me pregunto si Michiel de Vries (autor de Meridian) habría comenzado un driver para moonsound si yo no le hubiera presionado a hacerlo hace años, en una feria de Tilburg. Hay muchal diferencia con otros ordenadores donde hay riqueza en la elección de trackers.

Hubo también poca innovación al construir el wavekit por defecto para los ROM-tones con la única ayuda de tonos útiles y el resto siendo casi lo mismo. Naturalmente todo "tenía que ser" compatible GM o algo así (otra de las razones por las que yo odio el MIDI/GM ya que el mundo MIDI está lleno de "lemmings" todos caminando en la misma dirección sin importar lo que suceda). El GM fue un mal invento desde el punto de vista de un técnico de sonido. En los 80 había riqueza de sintetizadores que sonaban todos distinto... ¡perfecto! Realmente tenías una razón por la que escoger un modelo en concreto. Entonces llegó el GM y de repente todas las cosas llegaron a ser GM, echando por tierra las variaciones del pasado. De todas formas, la gente estuvo escuchando ficheros MIDI de la misma forma que escuchamos los MP3 hoy, por lo tanto hubo realmente una demanda de un estándar. Nunca comprendí porqué el MSX tenía que participar en la historia del General MIDI ya que todos veníamos del PSG, SCC, FMPAC y Music Module, nada que ver con la escucha de ficheros MIDI o la creación de los mismos. El chip OPL4 era, obviamente, un chip GM, nada que Sunrise pudiera cambiar. Al final el wavekit en ROM podría haber sido más práctico en vez de "compatible GM". Así pues, ¡vamos a fijarnos en Meridian!

[El nuevo siglo]

-La creación de Infinite en 2004 fue el principio de un éxito enorme en la "nueva" demoscene del MSX. ¿Cómo y por qué fue creado este grupo?

Cuando me mudé en 2002 acabé viviendo a unos 30 minutos del pueblo de Frederik Boelens (chaos), si tenéis los emails de la lista de correo de MSX podréis leer sobre esto. Así pues nos fuimos encontrando por diversas razones, como cuando viajábamos a una feria. También le ayudé a comprar un piano y las lecciones de piano y algunas veces se descargaba algo grande para mí ya que yo no podía hacerlo con mi línea telefónica entonces (me lo daba en la estación central de su pueblo). En uno de esos encuentros al final de 2002, él quiso hacer una demo para navidad y mientras permanecía en las escaleras que llevan al tren otra vez volvió a preguntarme si yo quería hacer una canción para ella. Eso marcó, eventualmente, el comienzo de mi reentrada en la "scene". Tras eso trabajamos en juegos que nunca despegaron. Al año siguiente el MRC tuvo la competición Snowfall en la cual chaos quería participar. Juntos comenzamos el concepto de la nieve cayendo dentro de las letras en scroll. Ayudé con mi camión de pequeñas utilidades hechas en Blitzbasic con las cuales se preparó la intro del campo estelar, el scroll en SC4 y las conversiones de gráficos. En ese momento pensamos en crear un nuevo grupo ya que Teddywarez estaba ya parado. Tras

algunas sugerencias para el nombre surgió "Infinite" y creo que ése fue el comienzo. Llegó el concepto del logo de plasma tras hacer un prototipo en Blitzbasic y ahora es parte de todas nuestras producciones (¡menos las de MSX1, obviamente!). Sea como fuere, fuimos terceros en la competición y eso no fue un mal comienzo. Medio año después vino la competición Underwater y en ese tiempo Albert Beevendorp (Bifi) se había unido a Infinite tras ser originalmente el colaborador de chaos en la programación. No tuve la intención de que fuese algo grande, así que comencé con una melodía fácil. Los 4 acordes sampleados en los primeros segundos son prueba de ello ya que tienen 32K cada uno, ¡algo desproporcionado! Normalmente cuando creo samples tan grandes al comienzo quiere decir que no voy a hacer nada grande, de otro modo habría hecho samples más pequeños para dejar espacio al resto. De todas formas, por alguna extraña razón la canción progresó bien. No creí que fuera necesario añadir muchos efectos demo (más que el logo "waterwave") por lo que quedó en un slideshow con un poema, un atrevido concepto si quieres competir en una demoscene donde los programadores típicos comen pantallas divididas. :)

Al final de 2004, Edwin apareció de nuevo y pronto le persuadí para entrar en Infinite. Él renovó el motor 4motion para que trabajara a 3.58MHz y poco después comenzamos Universe: Unknown. En verano de 2005 Infinite y Bifi se separaron y así somos 3 personas de nuevo. Sin embargo chaos necesita unas pocas patadas en el culo para que acabe algunos de los juegos en los que estuvimos trabajando. ¡Podéis colaborar en molestarle con eso!

-La gente habla de las últimas demos de Infinite. Explícanos cómo Infinite logró crear 4-Motion y cómo sincronizáis los gráficos y la música. Todo el mundo puede darse cuenta de esto cuando las bolas chocan en la demo "Sphere".

En un momento dado me di cuenta de que con 4 colores como máximo y 256 caracteres podías tener una resolución de 64x48 píxeles en screen 4 que podía ser actualizada en cada interrupción. Es sólo una forma lógica de pensar, no un concepto complejo. La cuestión aquí no es "¿puedes programarlo?" sino "¿puedes hacer una película con 4 colores y una resolución de 64x48?". Comencé en un editor donde colocar frames uno tras otro y editar la paleta. Fueron semanas para conseguirlo y el día de entrega me di cuenta de un desagradable bug que me tuvo hasta las 21:00. Desde ese momento todavía me quedaba editar la demo completamente (la música ya estaba hecha) y la fecha límite eran las 23:59. Conseguí un poco de tiempo extra por parte de Snout, pero ni por ésas. A las 03:00 Bifi (quien programaba todo por entonces) fue a dormir y yo continué toda la noche. El túnel púrpura, la ciudad y otras pocas partes fueron hechas esa noche. El primer (y hasta aquí el único usado) editor de 4motion tiene una configuración de tempo primitiva; estaba preparada para un tempo 19 en MBWave y con 30 fps querría decir que cada paso en MBWave tomaba 3 frames de la película. De esa forma la sincronización era fácil. Para Wings el editor se preparó para un tempo 17 (4 frames por paso). Más tarde hice un nuevo editor todavía no usado sin límites de tempo (siempre que se basen en VDP) pero nunca se ha utilizado ya que me aburrí de tener sólo 4 colores y películas de una única capa. En el momento en que una nueva 4motion demo aparezca, ¡podéis esperar a que será más avanzada! La canción para Sphere fue hecha antes que la demo. Generalmente me surgen imágenes cuando compongo y aquellas imágenes imaginarias me dan ideas para notas musicales nuevamente. Es un proceso de retroalimentación al crear música sin tener imágenes físicas.
La gente realmente habló sobre esto y de alguna forma creo que esta demo hizo que la gente fuera consciente de que está la "oldskool" y la "newskool". Quizás fuera el punto de vista donde la gente no viera demos de MSX como "demos de MSX" nunca más sólo como "demos oldskool en un MSX". Si miras atrás en los años de la demoscene y resumes lo que la gente hizo, verías scrolls de texto junto a olas, ciclos de colores, etc. La Unknown Reality de NOP fue una de las primeras demos, si no la primera, en abandonar ese concepto, aunque creo que críticos independientes la pondrían bajo la "oldskool" también. Bandwagon es también un buen ejemplo de nuevas iniciativas y es mucho más cercana a lo que ves en parties de demos que nada hecho anteriormente. No sólo en términos de efectos pero también en términos de estilo.



Sphere de Infinite (2005)

4Motion es, francamente, un reproductor de películas no lineal, aunque esto no es incluso "tan" difícil. El enfoque se mueve hacia el arte aunque es una buena noticia para los artistas quienes normalmente tenían que esclavizar a los programadores y a sus ideas.

¿El ingrediente básico para hacer una demo de MSX entretenida? Comienza con la música, haz un principio y un final en la canción sin aburridas repeticiones. Intenta darle una longitud de 4 ó 5 minutos, pon elementos coordinados en ésta (puntos

donde podrías ver algo cambiando en la pantalla) y entonces crea una demo sobre esa canción. Intenta "seguir la corriente de la canción" usando código y arte visual. Oh, y una cosa: los textos con scroll van fuera. :P Otro consejo: no pienses en la resolución del MSX y en los colores, sólo permite a los artistas hacerlo. En la vieja demoscene un simple bloque relleno en la pantalla sería una vergonzosa cosa que hacer, pero estos días... ¡inténtalo! Vamos a seguir la canción, ver qué puedes conseguir de ella. Tener cosas en alta resolución, con bordes alisados, suaves, etc. es todo de la vieja escuela y nosotros ya tenemos una pila de demos como ésas ya.

Echa un vistazo a las nuevas demos "newskool" de otras plataformas, creo que puede ser muy educativo y algunas veces ¡incluso sorprendente!

-Infinite nos sorprendió durante el último MSXDev con el juego Universe: Unknown. Un juego tipo Gradius con música PSG. La melodía es genial e impresionó el cambio conceptual: del "techno" en el Moonsound con 512KB de Sample RAM al estilo clásico de la música de videojuegos en PSG con 3 canales. ¿Cómo te sentiste al regresar al PSG?

En mis años de demoscene para PC siempre recibí críticas como... "por supuesto, tú y tus 100 canales y tropecientos samples", como si yo no pudiera hacer música con menos requisitios. Bien, aquí hay 3 canales con ondas cuadradas... ¡en vuestra jeta! :) En un momento puntual realmente me acostumbré a ello e hice 4 temas para Universe: Unknown en un día, tras deshacerme de 3 ó 4 intentos medio terminados. En términos de estilo hay dos formas en las que se puede trabajar respecto al PSG. Está la forma tipo PT3 con arpeggios, formas de onda en PSG, etc. y está la otra vía que es más cercana a Konami. Prefiero la segunda. Nunca he sido un fan de los arpeggios y similares ya que prefiero componer música de 3 canales que música de 5 canales donde 3 canales son el resultado de 1 canal de arpeggio. Es mi preferencia. Todavía creo que la música de Usas es incomparable en el mundo PSG así que para mí es una elección fácil entonces: ¡Universe: Unknown tendrá estilo Konami en las canciones PSG!

Ahora, desde que no considero mis primeras canciones en BASIC para PSG como música real, Universe: Unknown es mi primera música para PSG. Fue hecha en MBWave y por el momento estoy perfilando un nuevo formato para reproductor el cual soportará FMPAC, PSG y SCC. Ya probé este método volviendo a arreglar el nivel 7 del juego a FMPAC y PSG y los resultados son muy prometedores. La música podría entonces ser hecha en PC, lo que quiere decir que la última parte que tenía que ser siempre hecha en MSX también se mueve hacia el desarrollo cruzado. Sí, ¡es algo demoníaco! (pero se lo recomiendo a todo el mundo) :)

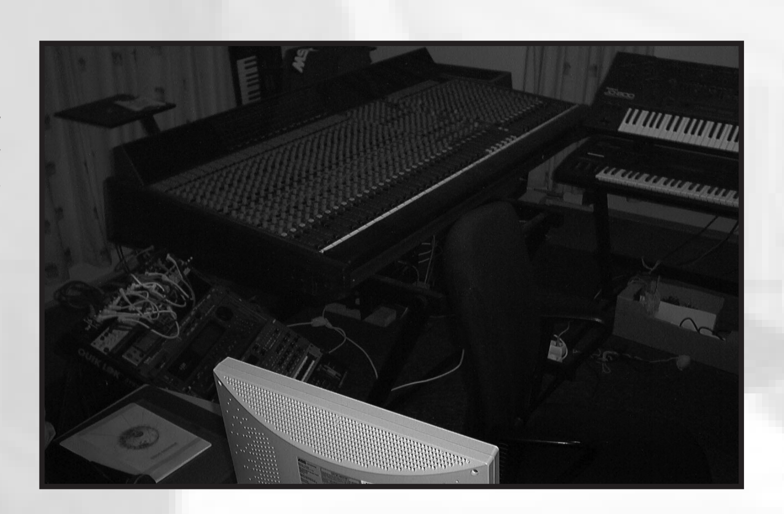
-Los usuarios de MSX recuerdan la música de los juegos antiguos, cosa que nunca sucede con los nuevos por el tipo de música ambiental que tienen. ¿Qué opinas de esto como compositor?

Si te estás refiriendo a los juegos de MSX recientes es cierto. Básicamente algo que dije antes: Creo que los juegos antiguos fueron hechos por compositores profesionales con una fuerte educación clásica. Otra teoría puede ser que los compositores de la scene raramente escuchaban bandas sonoras de películas y ellos tenían referencias limitadas a qué tipo de música encaja en un concepto visual. Uno de mis hobbies es coleccionar BSOs. He escuchado casi todo de John Williams, mucho de Jerry Goldsmith, Hans Zimmer, y también una decente cantidad de gente como James Newton Howard, Danny Elfman, George Fenton, Alan Silvestri, etc. Son todas referencias perfectas si quieres hacer música para películas o juegos. Pero como ya he dicho... esto es únicamente una teoría, quizás la música de los juegos de siempre de los 80 es únicamente arte que ha muerto.

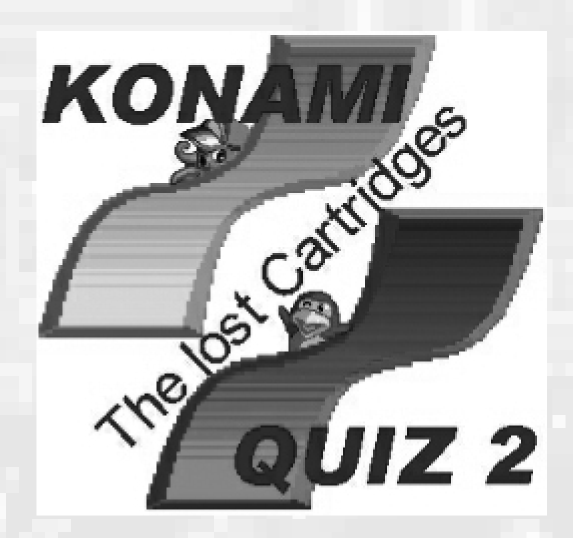
-¿Va Infinite a hacer demos o juegos para el GFX9000?

Sí, tanto Edwin como yo compramos GFX9000 en la feria de Nijmegen de 2006. Por supuesto nosotros necesitamos nuevas herramientas si estoy yendo a pixelar para GFX9000, por lo que esto requerirá esfuerzos extras. Estamos actualmente preparando algo ya, pero ¡vamos a mantenerlo como secreto! De todas formas, estamos preparados para el MSXDev'06 y como ya has podido adivinar llenaremos un megarom completo. Y esta vez quiero que esté hecho a tiempo para evitar el stigma de Infinite de lucha contra la fecha límite y la posterior negociación con los organizadores para esa hora extra de tiempo. ;)

PD. Ya hay un título preparado para el juego de la MSXDev'06: "title: unknown" ... ¡y eso es todo lo que el mundo va a conseguir de nosotros ahora!



Ya disponible



Bombaman fue la última producción holandesa que pudimos disfrutar y por fin vamos a tener en nuestras manos un nuevo título procedente de aquellas tierras que tanta gloria dieron durante los noventa a nuestro querido MSX. Konami Quiz 2, como su propio nombre indica estamos ante un juego de preguntas y respuestas inspirado en la mítica firma nipona y en sus creaciones y personajes que se han convertido ya en clásicos de los videojuegos.



La última producción de Delta Soft nos sorprende técnicamente al ponerse a la venta en una tarjeta Compact Flash. Nos parece una idea fantástica ya que así no sólo usamos el nuevo hardware como almacenamiento masivo sino como un soporte más a tener en cuenta como el cartucho o el propio diskette. Pero los usuarios que no tengan dicho soporte también pueden hacerse con este título en formato disco. Siete en total son los discos que forman este juego por lo que la comodidad de la CF vuelve a imponerse.

La historia del juego se resume en que Tako (un malvado pulpo) ha robado seis cartuchos a Peter Penguin mientras dormía y éste tratará de recuperarlos enviando una carta al periódico. En definitiva, dos personajes que van recuperando y perdiendo cartuchos durante todo el juego.

En éste podemos participar hasta cuatro jugadores, lo que lo hará más divertido. Los idiomas serán en este caso holandés e inglés. Así que si no vamos muy fuertes en éste último necesitaremos un diccionario. Hay que decir que los textos son de un inglés elemental, muy fácil de seguir.



Konami Quiz



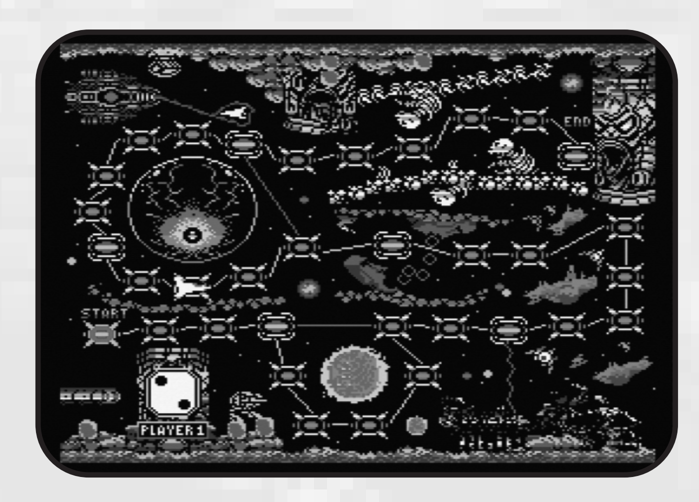
Find It



Thunderbirds are go

Delta Soft es uno de los grupos más antiguos que se mantienen en Holanda. Comenzaron a hacer cosas en el MSX en 1987 aunque nunca pasarán a la historia por la calidad de sus producciones. Hasta la fecha se han limitado a crear demos, juegos de puzzles y utilidades. Eso no impide que sean uno de los grupos que siempre están ahí sacando cosas con una actitud ejemplar. Su manera de hacer es capturar imágenes vía digitalización y lo mismo podríamos decir de la música, abusando de los samples del Music Module. Salvo alguna excepción en que se han aliado con Bi-Fi para la creación de músicas en Stereo y ya en Moonsound en la aventura gráfica Thunderbirds por Qix. En este nuevo título tenemos la novedad de que las músicas son en SCC.

Konami Quiz 2



Una vez iniciado el juego y seleccionado nuestro personaje nos irán dando a elegir entre varios tableros inspirados en los juegos de Konami para MSX. Una vez elegido uno deberemos pulsar la barra espaciadora para parar un dado y obtener un número que serán las casillas que avanzaremos. En cada casilla nos harán una pregunta que deberemos acertar. Al llegar al final del tablero y si acertamos la última pregunta conseguiremos recuperar un cartucho robado. Si juegan más de 2 jugadores podremos ir comprando objetos y participar en los minijuegos.

Estamos ante un juego que está entre el Trivial Pursuit y el Rune Master. Parece que este género siempre está de moda como podemos comprobar con la actual saga Mario Party, pero claro, salvando las diferencias. Aquí prima más las preguntas que el tablero en sí. Se puede reprochar que deberían haber más preguntas diferentes, ya que a veces se repiten. Hay cinco minijuegos que harán más variado el juego.

Los gráficos son decentes; todo lo que es ripeado tiene un aspecto excelente. Los gráficos propios son bastante más flojos. Las pantallas del tablero tienen mayor o menor gusto en cuanto a concepción. Algunos son una borrachera gráfica, muy confusos y otros son delicados y de bella factura como la pantalla que hace honor al F1-Spirit.

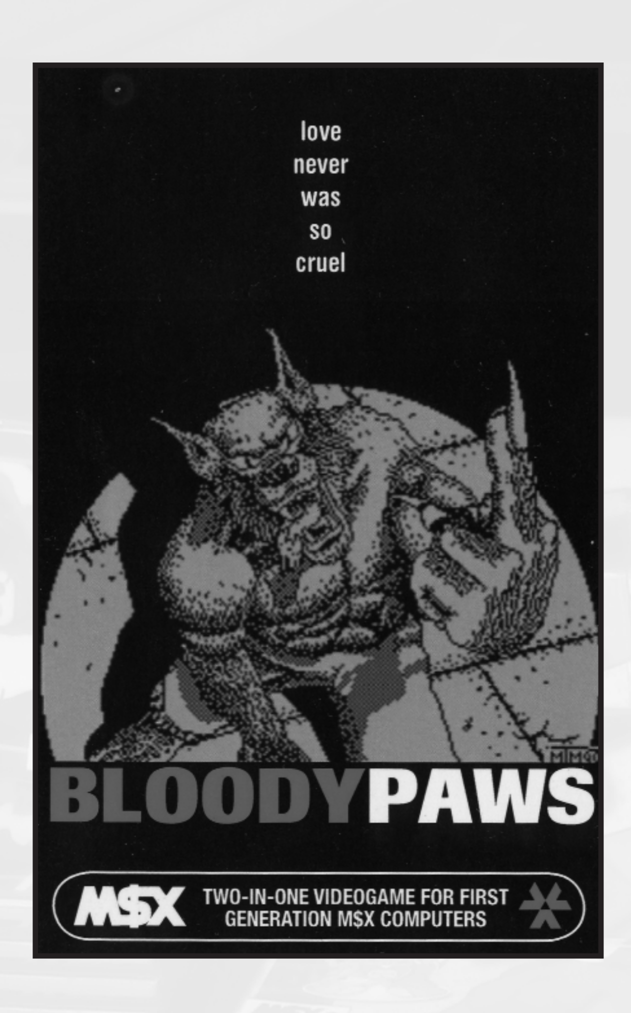
La música aprovecha el cartucho de sonido SCC y las melodías están en sintonía con el sonido Konami de toda la vida. Muchos temas son de cosecha propia y en otras se adaptan temas, es decir, te suena mucho la canción del juego en cuestión pero es otro tema diferente. Nos sorprende que no hayan añadido efectos de sonido, pues creemos que los necesita (sonidos de los dados, de los avances, al acertar o fallar una respuesta ...). Incomprensible.



Para disfrutar más el juego (mayor duración) es conveniente que jueguen hasta 4 jugadores a la vez. Así se hace más difícil conseguir los 6 cartuchos necesarios para finalizar el juego. Tú decides, o te montas una orgía o enciendes el MSX.

Aunque el juego en sí no es una maravilla, hay que reconocer el esfuerzo en su realización, ya que es bastante laboriosa su concepción. Técnicamente no es nada del otro mundo pero es una buena oportunidad de aprender hasta los mínimos detalles de la mejor compañía que ha tenido el MSX en su historia. Saldréis hechos unos expertos en la materia, unos jodidos freaks.

Jugabilidad:	6
Música:	8
SFX:	0
GFX:	8
Duración:	8
Extras:	6
Total:	**6**



Un juego que llevaba 15 años en la oscuridad. Y es que este juego fue publicado en Spectrum sin el consentimiento de sus creadores por la compañía GLL Software en 1990. Tras la sugerencia de Matra, Juan Carlos Sánchez, su creador, finalmente optó por sacar a la luz la versión de MSX. Así en la Reunión de Barcelona pudimos comprar un juego en formato cassette de 64k como lo hubiésemos hecho 15 años atrás. Parece que el tiempo cura las heridas y por fin tenemos un título más en la ludoteca MSXera.



Así pues con esta historia para no dormir se nos presenta este Bloody Paws, el cual lo tenemos que presentar como es, un juego de hace 15 años, y en la informática, es mucho tiempo; por lo cual hemos de valorarlo siempre pensando que no es un juego de reciente factura, sino con las limitaciones de aquella época, aunque la máquina sea la misma no lo es el concepto sociocultural de esa fecha.

En esos años estaba muy de moda los beat'em up, podemos poner muchísimos ejemplos (Final Fight, Dragon Ninja, Double Dragon y un largo etcétera). Podríamos acotar esos años entre 1987 y 1992. Quizá fue el Street Fighter la que acabó con esta moda de peleas callejeras en los suburbios de las grandes ciudades. Todas las firmas sacaban juegos de este tipo con más o menos éxito, pero siempre en máquinas de 16 bits. A finales de la década de los ochenta todavía no estaban asentadas las consolas por España y los ordenadores de 8 bits daban sus últimos suspiros. Hacían versiones cutres prácticamente nulas de color (recordemos siempre las buenas fotos a todo color al dorso de la caja de las versiones Amiga o Atari) para los cuatro sistemas decadentes de siempre. Era la época de los After The War y compañía. De esta forma podemos entender el contexto de este juego. Un programador y unos grafistas se animan a programar un juego para Spectrum, dado que era el sistema donde siempre se editaban la mayoría de títulos en cassette. Y la cosa no sale como esperaban.

Después de 15 años de silencio tenemos un videojuego arcaico en nuestras manos y nos produce una extraña sensación. Un juego de MSX arcaico, en cassette y con su estuchito típico que hacía siglos que no veía. Ya no recordaba lo del Run"Cas:. Parece mentira que encuentros así sacudan la memoria de una persona. Sí señor, una sensación fuerte.

Y ya de lleno en el juego decir que estamos ante un juego de MSX1. Sí, en principio pueden jugar casi todos los usuarios que no tengan un Turbo-R. ¡Qué cosas tiene la vida! El mejor ordenador de la norma no puede cargar cintas. Con un simple MSX1 de 64k podremos jugar si tenemos una lectograbadora. Debemos pulsar control y shift mientras encendemos el ordenador para disponer de la máxima memoria para ejecutar el juego.



Bloody Paws





Durante la carga nos aparece un dibujo correcto en Screen 2. Ya a la hora de jugar pulsamos espacio y llevamos a un hombre lobo por las calles de una ciudad buscando a su amada. Por la travesía nocturna iremos golpeando con nuestros puños a todo aquél que se nos arrime con malas intenciones. Nos vendrán incluso "volando". Podemos saltar y dar golpes de pie o agachados.

El control no es muy bueno que digamos. Los movimientos no parecen responder con gran fiabilidad, lo que lo hace un programa difícil, para desesperarse.

En cuanto al grafismo, podemos decir que es lo mejorcito del juego. Estamos ante unos gráficos correctos obra de MAD & Matamoros, no son nada espectaculares, pero tienen un tamaño considerable y eso pocas veces lo hemos visto en nuestro ordenador. Y si además tenemos scroll pues los parpadeos son más que constantes. Cromáticamente hablando es escaso, ya sabemos de sobras cómo son las adaptaciones del hermano Spectrum. Quizá lo más sobresaliente sea el dibujo de los diferentes elementos de la vía pública tales como cabinas telefónicas, papeleras ... y en la segunda fase de la primera parte están muy logradas las casas y algunos árboles.

La música no existe, no hay melodía en el juego. Tan sólo disponemos, menos mal, de efectos de sonido, que tampoco son muy numerosos. Las pisadas, los puñetazos y cada vez que recibes daño de los enemigos y poca cosa más.

Dentro de lo sorprendente, aunque tampoco lo es tanto, ya que fue otra moda de aquella época era jugar en la segunda cara de la cinta a otra parte del juego mediante la obtención de un password al pasarse el primero con anterioridad. Pero viendo lo difícil que parece conseguir tal objetivo pocos serán los que logren dar la vuelta a la cassette si no ven el truco publicado en algún medio informativo.

La segunda parte, es decir, la cara 2 es más de lo mismo cambiando lógicamente el decorado. Ahora nos sitúa en un paisaje ruinoso con esta-tuas y gárgolas y con unos enemigos terroríficos, ya sean pajarracos prehistóricos, calaveras ... Tan difícil de jugar como la primera cara.

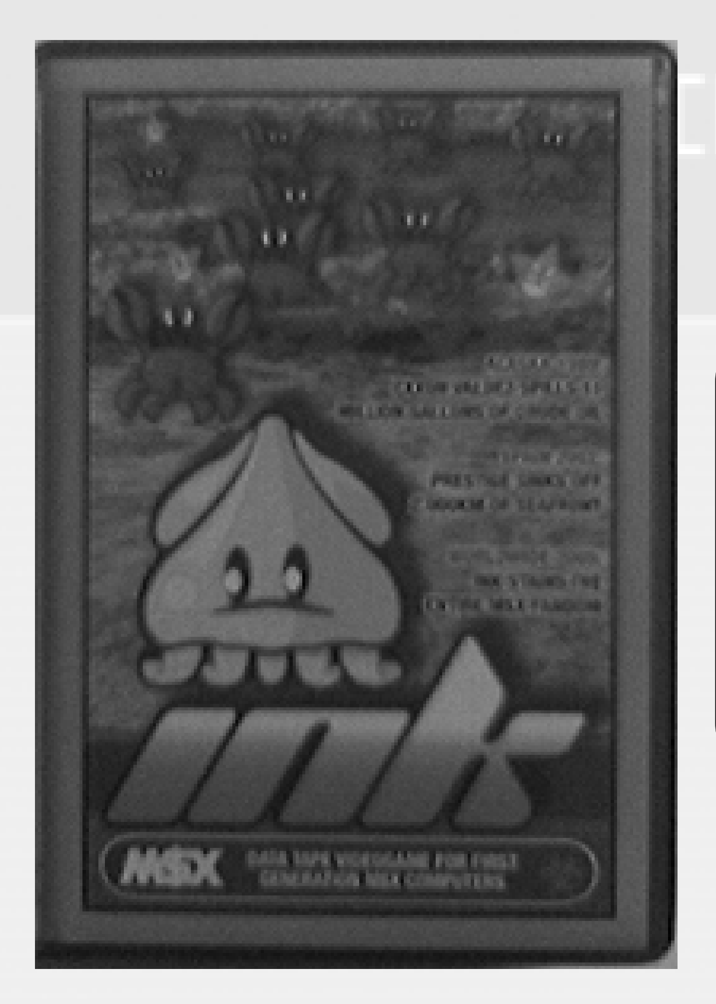
Jugabilidad:	6
Música:	0
SFX:	6
GFX:	7
Duración:	7
Extras:	5
Total:	**6**

En fin, un juego que en su época ya hubiera sido del montón, que hubiera pasado con más pena que con gloria por el mundillo de los videojuegos, pero que será recordado por motivos ajenos a lo que la programación se refiere.

Si eres un fanático de los juegos de cinta o un coleccionista incompulsivo seguro que te harás con él, pero si en cambio eres de los que busca un juego 10, éste no es el mejor ejemplo que digamos.

Ya disponible

Este es el segundo juego que presentó Matra en la pasada Reunión de Usuarios de MSX de Barcelona. Estamos ante otro juego en formato cassette, de 32k de memoria, que por sus características podría haber participado perfectamente en el concurso Msx-Dev pero ha optado por no participar y venderse como producto suelto en un estuche para cintas y con una estupenda presentación.





Una sorpresa muy agradable por parte de un desconocido grupo llamado Ink Team. ¿Será un pseudónimo de algún miembro de Matra? No lo sabemos, tiempo al tiempo. El caso es que ha sido Matra el encargado de empaquetarlo como se merece, con un estuche típico de cintas de los ochenta muy bien maquetado, con un nivel de profesionalidad envidiable.

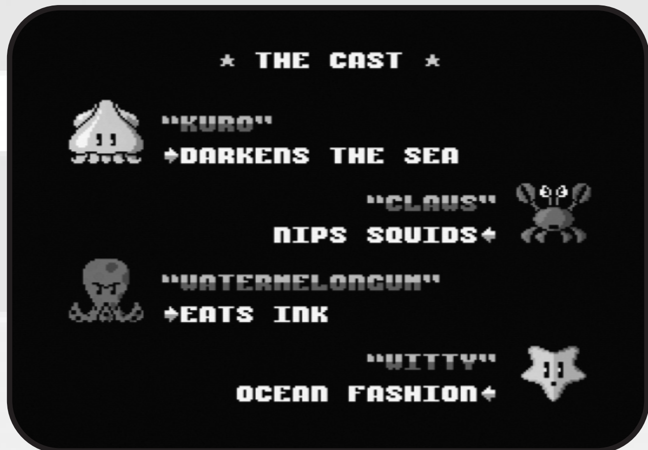
Pues es una sorpresa como decía antes disponer de este juego en formato cassette y no en disco como nos tenía Matra acostumbrados unos añitos atrás. Pero aquí los experimentos están a la orden del día. Y piensan atacar próximamente en formato cartucho. Pero bueno, lo importante no es el medio sino el fin, y lo que tenemos ante nosotros es un juego nuevo muy bien hecho, de los que crean afición. Puede ser que la cinta pueda ser un formato que llegue a más usuarios y eso sea una visión mercantilista muy digna. Hay que pensar que siempre hay un sector de usuarios que sólo les gusta su MSX1 con cintas porque es como recuerdan una época de su infancia y los ordenadores posteriores no les interesan. Tiene que haber de todo en la viña del señor.

La primera impresión al ejecutar el programa es muy buena, parece la típica rom japonesa de toda la vida. Una presentación muy digna con un gráfico del personaje protagonista grande y divertido. Si lo dejamos pasar un rato aparece otra pantalla donde aparecen el resto de personajes del juego mientras éstos están animados. Es un homenaje a los videojuegos clásicos, donde era muy común especificar en una pantalla los enemigos que te ibas a encontrar y la puntuación por deshacerte de ellos.

Llevamos al calamar Kuro quien debe ensuciar con su tinta el océano mientras el cangrejo Claws intentará impedírtelo, al igual que el pulpo Watermelongum, quien irá limpiando tu tinta. Para más inri la estrella de mar Witty nos hará de obstáculo en nuestro camino. Estamos ante un juego clon del mítico Amidar y de tantos otros que salieron siguiendo su estela. Pese a lo poco original del concepto de juego sí que la historia es diferente a mucho de lo visto.

El juego no es tan fácil como parece, no es sortear a los enemigos y rellenar las casillas. Hay más dificultades de las que parecen a simple vista. La tinta se agota, por lo que debemos volver a zonas que ya hayamos manchado antes para recuperar nuestro nivel de suciedad. Si nos vemos muy apurados por los enemigos tenemos la posibilidad de pulsar espacio y durante un instante los enemigos quedarán paralizados. Decir que esta opción sólo funciona tres veces en un mismo nivel, así que los recursos no son eternos.

Los gráficos son convincentes, no hay nada que llame la atención. Eso quiere decir que no hay desequilibrios gráficos, todo está en la misma coherencia gráfica. Esa es una buena

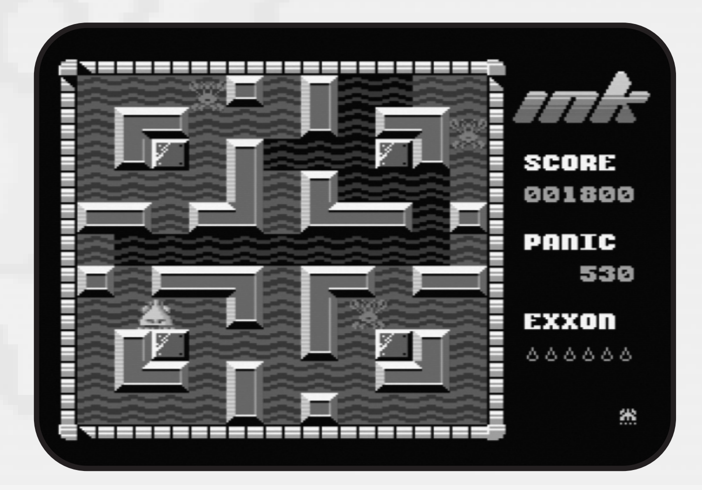


Jugabilidad:	9
Música:	8
SFX:	8
GFX:	8
Duración:	8
Extras:	6
Total:	**8**

señal porque transmite una sensación de profesionalidad. Todo es muy equilibrado, el diseño, los espacios. También debemos destacar el efecto de scroll del agua que sube y baja en un segundo plano. Ink ha sabido emular el concepto clásico de videojuego con todas sus consecuencias.

La música es Psg. Si sabes aprovechar bien los tres canales de sonido se puede tener una banda sonora digna. Este juego tiene melodías correctas (la inicial es una famosa versión de un tema de los Ramones), eso sí, siempre entre fases, porque mientras jugamos sólo oímos efectos de sonido que imitan las olas del mar. Creo que han acertado porque no queda nada mal este resultado, ya que al tener más efectos de sonido bien logrados te metes de lleno en la partida.





La jugabilidad es lo mejor de este juego. El personaje se mueve muy bien y no hay brusquedades en ningún momento, es todo de una suavidad deliciosa. Además hay que contar con una adicción notable, siempre apetece hincarle el diente a este Ink. Lograr que te enganche es uno de los propóitos más complicados de un videojuego, y este juego lo hace. ¿Será porque no siempre tienes que hacer el mismo recorrido? ¿O será por la dificultad que va in crescendo? El caso es que este juego tiene todos los alicientes para hacerte con él. Sencillo pero muy bueno. Altamente recomendable.

MÁS SORPRESAS

Por los rumores que se cuecen parece que va a salir quizá para la próxima Reunión de Barcelona una nueva versión de este juego aprovechando las características propias de los ordenadores de segunda generación. Y lo más novedoso es que podría tratarse del primer juego en formato cartucho desde tiempos immemoriables. Puede ser un bombazo el disponer nuevamente del soporte cartucho para disfrutar de un juego como éste y más para MSX2. Lo que no sabemos es si podremos disfrutar de esta misma versión MSX1 en esa misma Rom.

Así que parece que va a valer la pena acercarse nuevamente a las Cocheras de Sants a finales de Abril para hacerse con una copia de un juego tan digno como este Ink.

Y sobretodo, deseamos saber quién se esconde tras el nombre de Ink Team, pues nos tiene a todos intrigados. ¿Se descubrirá en la Reunión de Barcelona o por el contrario seguirá creando juegos desde el anonimato?

Remakes de juegos MSX

Lo último en software independiente y (II)

Una vez mas seguimos comentando los remakes existentes de esos juegos que marcaron nuestra juventud.

ABU SIMBEL PROFANATION - ABU SIMBEL PROFANATION DELUXE
(M.A Software)
http://194.224.229.190/masoft/index.html



Este remake se presenta en 2 versiones, una normal y otra deluxe, ésta última con mejores gráficos.

Este juego era uno de los más difíciles a los que tuve la ocasión de jugar, ya que si no calculabas bien cuando y desde donde saltabas te ibas a la tumba directo.



La principal novedad es que presenta más colorido y música mientras jugamos, pero al mismo tiempo comparte los mismos errores del original: no permite grabar la partida y, en algunas pantallas, salir de la pantalla por la salida equivocada significa ver como mueres una y otra vez hasta agotar todas tus vidas. Por lo demás es tan jugable como el original de Dinamic.

ALPHAROID *(Benjamin Aeilkema)*
http://www.remakes.org/

Este remake poco tiene que ver con el original de MSX, para los que no conozcáis el original, el juego se divide en 2 etapas; en la primera llevas un soldado volando y disparando en un scroll horizontal en plan Nemesis, durante esa fase te tenías que ir fijando en el suelo para que cuando vieses una abertura en el suelo te colases por ella para acceder a la fase a pie y liquidar al jefe de turno al estilo Double dragon.



Para este remake los programadores han cambiado algunas cosas:

-Para empezar el escenario no ocupa toda la pantalla lo cual disminuye enormemente el área de juego.

-Si bien la primera etapa presenta un scroll a 3 bandas y enemigos muy currados gráficamente, la parte de scroll que representa el suelo es muy lento y no hará falta de que os molestéis en buscar aberturas en él para ir a buscar a los enemigos, al cabo de mucho rato matando bichos en plan Nemesis el scroll se detiene y aparece un cuadrado dando vueltas, entrad en él para acceder a la fase a pie.

-Para la etapa a pie tampoco os molestéis en buscar un guardián espectacular, tan sólo unos pocos malos caminando y pegando patadas.

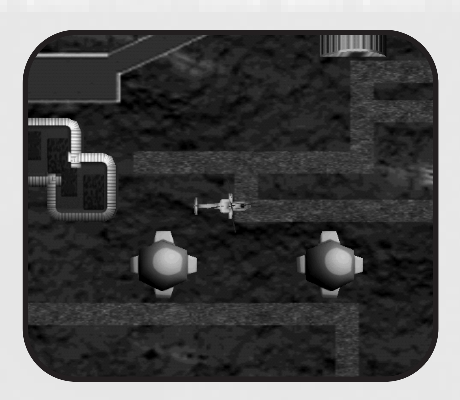
Dejando de lado esto, el juego presenta excelentes gráficos de los enemigos, y músicas en midi de Jan Van Valburg de varios juegos de MSX, sin embargo falla en cuanto a la jugabilidad siendo la etapa primera demasiado larga.

RAID ON BOTANY BAY (RAID ON BUNGELING BAY)
http://www.spacialfx.com

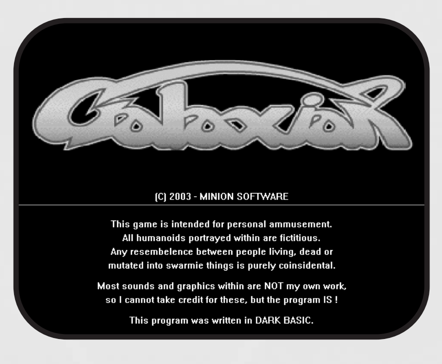
La mayoría de aquellos que en sus años mozos compraron un Sony Hit-Bit de seguro que encontraron este cartucho con el lote de juegos que lo acompañaba. Por si no lo habéis jugado nunca, tienes que manejar un helicóptero por un archipiélago destruyendo unas fábricas y, de paso, evitando que se termine de construir un barco de guerra en los astilleros de una de las islas.



El remake le hace justicia al juego y mantiene el mismo nivel de jugabilidad, si bien los gráficos al haber aumentado de tamaño respecto al original podrían estar un poco mejor.



GALAXIAN
http://www.minionsoft.com



Otro remake de uno de los juegos más adictivos que he tenido el gusto de jugar. Galaxians fue, en su momento, uno de los pocos clones de la recreativa Space Invaders que apareció en MSX, ya sabéis, oleadas de enemigos en formación que te atacaban sin piedad mientras caían hacia ti oscilando de derecha a izquierda.



El remake mantiene los gráficos originales del juego y su altísima jugabilidad. Aparte de eso poco más aporta, tan sólo mejores explosiones y efectos de sonido hechos para hacer retumbar los altavoces.

Lo dicho, si os gustó el original ya tardáis en bajároslo, pero recordad: seguro que no podéis echaros solamente una partida.

MOON PATROL
http://www.minionsoft.com



Otro juego basado en una recreativa, Moon Patrol apareció en una versión MSX-1 de la mano de Irem pero esa versión era, para que negarlo, bastante cutre, sus gráficos parecían dibujados por críos de parvulario.

Por suerte este remake sí hace justicia a la recreativa, al igual que en la misma lo peor es su música que a los pocos minutos se vuelve muy repetitiva, pero por lo demás volverás a terminar deseando volverte bizco para poder ver los baches que tienes que saltar al tiempo que intentas esquivar las bombas que te tiran los enemigos voladores.



FLOBO PUYO
(PUYO PUYO)
http://www.ios-software.com



De la mano de la gente de los Software nos llega este remake del juego Puyo Puyo. Otro de los cientos de variantes del clásico Tetris y en el cual recuerdo haberme quemado una pasta brutal en su versión de máquina recreativa.

Este remake no se inspira en la versión de MSX ya que no te deja jugar en solitario. En la opción de 1 jugador juegas contra la máquina o contra adversarios diferentes a pantalla partida y en dificultad ascendente, dispones también de 3 niveles de dificultad ajustables, también dispones de la clásica opción para 2 jugadores con la que te puedes echar unos piques con otro colega.

Tanto la jugabilidad como la dificultad se mantienen respecto al original, lo cual garantiza horas de diversión.

En cuanto a gráficos el juego está correcto y como punto negativo la música acaba por cansar al cabo de un rato de jugar.

Jordi Tor

COMPRESIÓN HUFFMAN

Por Avelino Herrera

En este artículo nos introduciremos en el apasionante mundo de la compresión de datos de la mano de un algoritmo clásico como es el de Huffman que, por ser clásico, no deja de ser uno de los más eficientes a efectos de velocidad aunque su capacidad de compresión ya haya sido sobrepasada por otros algoritmos. Este artículo está enfocado desde un punto de vista pedagógico y espero que sirva para despertar el interés de los programadores de MSX en este mundo.

En 1952 David A. Huffman publica una investigación en la revista "Proceedings of the I.R.E." titulada "A method for the construction of minimum-redundancy codes" (puede ser descargado íntegramente de http://compression.graphicon.ru/download/articles/huff/huffman_1952_minimum-redundancy-codes.pdf). Dicho trabajo de investigación marcó un antes y un después en las técnicas de compresión de datos. El algoritmo que se describe en este artículo, popularmente conocido como algoritmo de Huffman es la base de la mayoría de los actuales algoritmos de compresión existentes tales como el ZIP o el GZIP.

UN POCO DE TEORÍA

Para empezar diremos que existen dos tipos de códigos, los de longitud fija y los de longitud variable. Una codificación de longitud fija asigna la misma longitud de código para cada símbolo de un alfabeto: si, por ejemplo tenemos el alfabeto ASCII del MSX y trabajamos con base binaria (en este artículo, lógicamente, siempre hablaremos en base binaria) la longitud de cada código será de 8 símbolos o bits. La idea detrás de la compresión Huffman es asignar codificaciones de longitud variable en función de la probabilidad de aparición de cada símbolo del alfabeto. Consideremos la siguiente cadena de texto:

"ojo con el cojo"

Si utilizamos codificación ASCII del MSX lo que tenemos son 15 bytes. Analicemos las probabilidades de cada carácter:

'o' = 5 / 15 = 0.3333
' ' = 3 / 15 = 0.2
'j' = 2 / 15 = 0.1333
'c' = 2 / 15 = 0.1333
'n' = 1 / 15 = 0.0667
'e' = 1 / 15 = 0.0667
'l' = 1 / 15 = 0.0667

Como podemos apreciar la letra 'o' es la que aparece con mayor frecuencia, seguida del caracter espacio en blanco. Definamos ahora un concepto que nos será de gran ayuda: la longitud media de código

*L = sum(P(x) * l(x)) para todos los símbolos x del alfabeto*

Siendo P(x) la probabilidad de aparición del símbolo x y l(x) la longitud en símbolos de código (en nuestro caso bits) para ese símbolo x. Para códigos de longitud fija, l(x) es constante para todos los símbolos del alfabeto. Por ejemplo, si estamos hablando de texto l(x) = 8 siempre, mientras que si estamos hablando de pixels en una imagen de 16 colores l(x) = 4. Como la suma de todas las probabilidades es siempre 1 tenemos que:

L = l(x) para cualquier x

Para nuestro caso particular la longitud media de código es 8 bits. Ya que todos los símbolos del alfabeto (ASCII MSX) tienen una codificación de la misma longitud (8 bits).

Una primera aproximación a una codificación eficiente podría ser el asignar a cada carácter una codificación cuya longitud sea inversamente proporcional a su probabilidad de aparición. Por ejemplo, podríamos hacer la siguiente asignación:

'o' = 0
' ' = 1
'j' = 01
'c' = 10
'n' = 11
'e' = 001
'l' = 010

Una codificación de estas características tiene un gran problema: no podemos discernir cuando acaba un código y

comienza el siguiente:

'ojo con el cojo' ==> 0 01 0 1 10 0 11 1 001 010 1 10 0 01 0 = 001011001110010101100010

Si guardamos esta secuencia en un fichero o en memoria la guardaríamos así: 00101100 | 11100101 | 01100010, lo cual ocupa 3 bytes en lugar de los 15 bytes anteriores. Sin embargo, cuando queremos obtener el texto original nos encontramos con el problema que comentábamos antes:

00101100111001010110001 puede interpretarse como 001 0 1 1 001 11 10 0 1 ... que da como resultado 'eo enco ...'

De hecho se puede interpretar de muchas formas teniendo en cuenta la tabla de correspondencia entre símbolos del alfabeto (caracteres) y símbolos código (codificación en bits). ¿Por qué ocurre esto? La razón es muy sencilla: hay codificaciones que son prefijo de otras, por ejemplo, la codificación 1 correspondiente al símbolo espacio en blanco del alfabeto, es prefijo de las codificaciones 10 y 11 correspondientes a los símbolos 'c' y 'n', luego, cuando leemos un 1 en la entrada no sabremos si se corresponde con un espacio en blanco o con el primer bit de una 'c' o una 'n'. Para solucionar este problema tenemos los códigos prefijo o códigos instantáneos.

Un código prefijo o instantáneo es un código que asigna a cada símbolo del alfabeto, una codificación que nunca es prefijo de otra codificación del mismo alfabeto. Por ejemplo, si utilizamos la siguiente codificación:

'o' = 0
' ' = 10
'j' = 110
'c' = 1110
'n' = 11110
'e' = 111110
'l' = 111111

Lo que obtenemos es un código instantáneo ya que ninguna codificación es prefijo de ninguna otra. Con esta codificación del alfabeto, si escribimos la frase original 'ojo con el cojo' obtendríamos la siguiente configuración de bits:

'ojo con el cojo' = 0 110 0 10 1110 0 11110 10 111110 111111 10 1110 0 110 0 = 01100101|11001111|01011111|01111111|01110011|0 0

En lugar de los 3 bytes de antes, nos ocupará 6 bytes pero ahora obtener el texto original no ofrece ambigüedad ya que hemos utilizado un código instantáneo. Si calculamos la longitud media de código obtenemos:

*L = (0.3333 * 1) + (0.2 * 2) + (0.1333 * 3) + (0.1333 * 4) + (0.0667 * 5) + (0.0667 * 6) + (0.0667 * 6) = 2.8* bits por símbolo.

2.8 bits por símbolo y 15 símbolos hacen un total de 42 bits (15 * 2.8) que es la cantidad de bits que nos ha generado, en efecto, esta codificación para la entrada 'ojo con el cojo'. El algoritmo Huffman es un algoritmo que asigna de forma sistemática una codificación instantánea a cada símbolo del alfabeto de tal forma que es la codificación más eficiente y la que genera la menor longitud media de código posible para una entrada dada.

EL ALGORITMO

El algoritmo de compresión lo primero que hace es construir un árbol binario anotado a partir de la entrada en el cual, los nodos hoja representan los símbolos del alfabeto (ASCII MSX en nuestro caso particular). Veamos con el ejemplo de la frase anterior cómo generaríamos este arbol binario:

'ojo con el cojo'

Si tenemos N símbolos diferentes en el alfabeto de entrada, construimos N árboles con un único nodo y anotamos en cada nodo raiz el número de veces que aparece el símbolo del alfabeto asociado (su frecuencia entre paréntesis). En nuestro caso:

'o'(4), 'j'(2), ' '(3), 'c'(2), 'n'(1), 'e'(1), 'l'(1)

Figura 1

Consideramos que tenemos 7 árboles binarios con un único nodo raiz en el que anotamos la frecuencia de aparición del símbolo del alfabeto asociado. A continuación ordenamos los árboles por frecuencia:

'n'(1), 'e'(1), 'l'(1), 'j'(2), 'c'(2), ' '(3), 'o'(4)

Figura 2

Fusionamos los dos primeros árboles (o nodos) en un único arbol cuyo nodo raiz no tendrá símbolo asociado pero sí una frecuencia que será la suma de la frecuencia de los nodos raiz de los dos árboles que fusionamos. Es decir, hacemos esto:

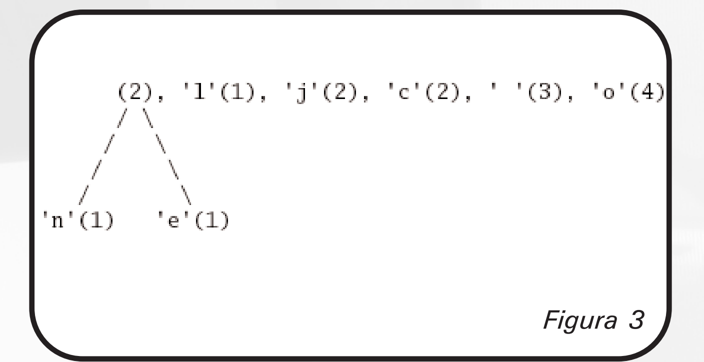


Figura 3

Hemos creado un pequeño árbol binario al fusionar los símbolos 'n' y 'e' y hemos anotado como frecuencia en el nuevo nodo raiz, la suma de las frecuencias de 'n' y de 'e' (1 + 1 = 2). A continuación reordenamos de nuevo los árboles en función de la frecuencia de su nodo raiz:

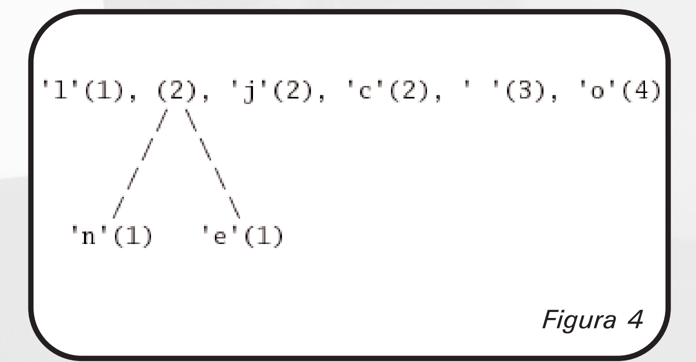


Figura 4

Volvemos a fusionar los dos árboles con menor frecuencia en un nuevo árbol y sumamos las frecuencias en el nuevo nodo raiz:

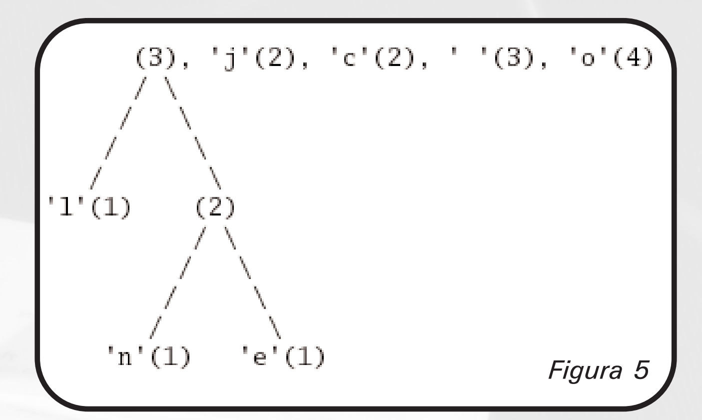


Figura 5

Reordenamos por frecuencia:

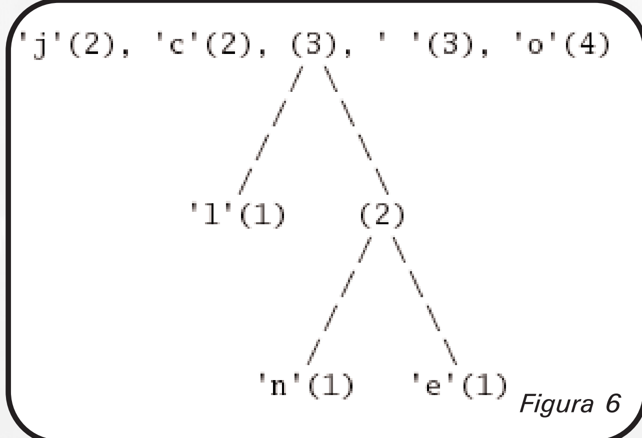


Figura 6

Y así sucesivamente. La construcción del árbol Huffman terminará cuando nos quede un único árbol ya que el algoritmo, en cada paso, disminuye el número de árboles en 1. Para nuestro ejemplo, el árbol Huffman resultante quedará como sigue:

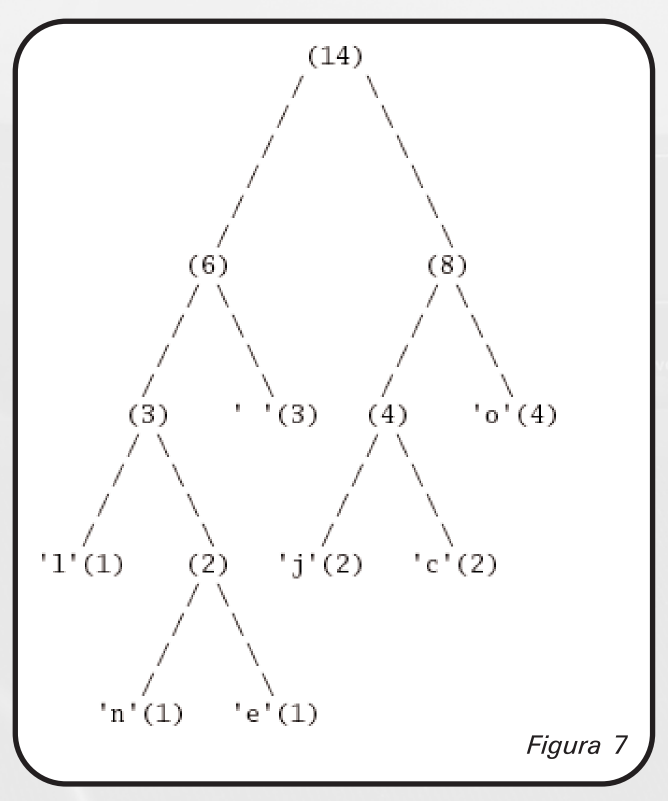


Figura 7

Como podemos ver los nodos terminales siempre representan símbolos del alfabeto de entrada. El procedimiento de codificación es muy sencillo: consideramos que las bifurcaciones a la izquierda simbolizan un 0 y que las bifurcaciones a la derecha simbolizan un 1 y etiquetamos todos los nodos en consecuencia (esta asignación es arbitratia y es igualmente válido asignar un 0 a las bifurcaciones a la derecha y un 1 a las bifurcaciones a la izquierda):

Compresión Huffman

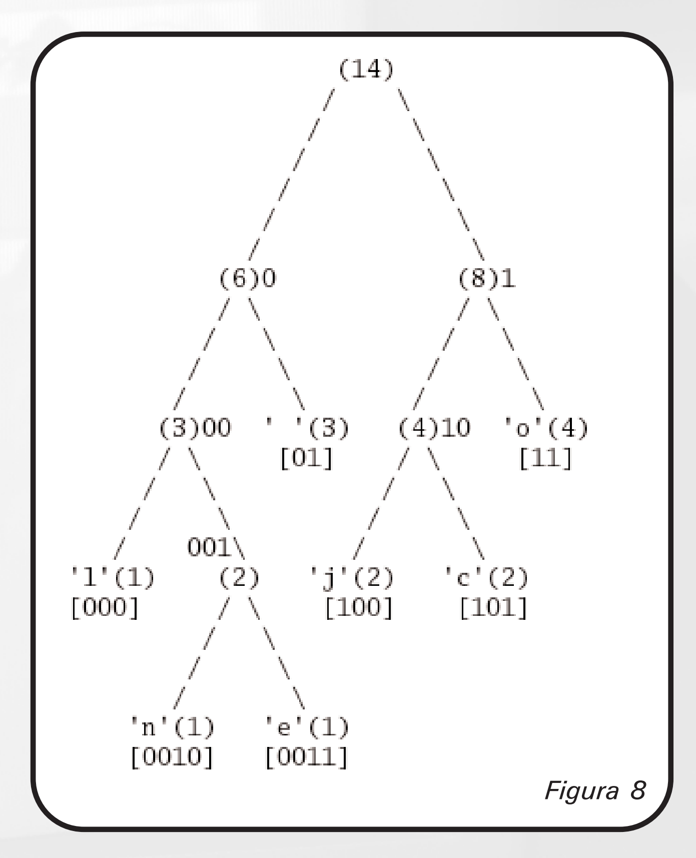


Figura 8

En corchetes podemos ver la codificación asociada a los nodos hoja del árbol:

'o' = 11
' ' = 01
'j' = 100
'c' = 101
'n' = 0010
'e' = 0011
'l' = 000

Y esta es la codificación que usaremos para comprimir la entrada inicial:

'ojo con el cojo' = 11 100 11 01 101 11 0010 01 0011 000 01 101 11 100 11 = 11100110|11011100|10010011|00001101|1110011

En este caso obtenemos una codificación que sólo ocupa 39 bits y que, por definición, es la más eficiente y la que genera la menor longitud media de código para la entrada 'ojo con el cojo':

*L = (0.3333 * 2) + (0.2 * 2) + (0.1333 * 3) + (0.1333 * 3) + (0.0667 * 4) + (0.0667 * 4) + (0.0667 * 3) = 2.6* bits por símbolo.

En efecto, 2.6 bits * 15 símbolos = 39 bits en total. Como se ve, el procedimiento de compresión es sumamente sencillo y el único handicap que tiene este algoritmo es que es preciso almacenar una representación del árbol Huffman ya que es necesario para poder descomprimir adecuadamente la ristra de bits comprimidos. Como se puede apreciar, el código resultante sigue siendo instantáneo.

Una vez cargado el árbol Huffman la descompresión es muy sencilla y rápida ya que lo único que necesitamos es, a medida que vamos leyendo bits, ir recorriendo el árbol y cuando alcanzamos un nodo hijo, emitir el símbolo del alfabeto asociado al nodo hoja y comenzar de nuevo en la raiz del árbol.

LA DESCOMPRESIÓN EN EL MSX

He implementado en C para el MSX (utilizando la sintaxis del compilador SDCC) el algoritmo de descompresión Huffman y el algoritmo de compresión está hecho en ANSI-C. Lo he hecho así pensando más bien en el desarrollo de juegos ya que normalmente lo que necesitaremos será que la aplicación (el juego) extraiga los datos (texto, pantallas, canciones, etc) de un fichero comprimido en disco. En http://msx.gabiot.com se puede obtener un fichero huffman.tar.gz que contiene la versión ANSI-C de los algoritmos de compresión y descompresión y, en una subcarpeta llamada 'msx' la versión para MSX del algoritmo de descompresión. Los ficheros fuentes compilan directamente en Linux y, en principio, pueden ser compilados y utilizados siempre y cuando la arquitectura sea Little-Endian (x86, ARM, etc) ya que el SDCC genera código que trabaja con datos Little-Endian para el backend de Z80.

El programa compresor 'hufenc.c' genera en Linux un fichero comprimido en formato HUF (lo he llamado así, no es ningún estándar) y este fichero tiene el siguiente formato:

Offset	Tamaño	Contenido
0	3	literal 'HUF'
3	2	offset en el fichero del nodo raiz del árbol Huffman
5	2	offset en el fichero del comienzo de los datos

Cada nodo en el fichero es un bloque de 5 bytes estructurados de la siguiente manera:

Offset	Tamaño	Contenido
0	1	dato (símbolo del alfabeto de entrada), este campo no se usa si no estamos en un nodo hoja
1	2	offset en el fichero del nodo hijo izquierdo
3	2	offset en el fichero del nodo hijo derecho

A partir del offset donde comienzan los datos comprimidos tenemos:

Offset	Tamaño	Contenido
0	4	entero de 32 bits que indica el tamaño en BITS de los datos comprimidos
4	??	los datos comprimidos en sí

Debido a que los offsets se indican en este formato HUF con 2 bytes, los nodos del árbol Huffman estarán al principio del fichero inmediantamente después de la cabecera del fichero, para evitar desbordamientos y offsets superiores a 32 KBytes. En el compilador SDCC (los enteros de 16 bits son de tipo 'short' y los de 32 bits son de tipo 'long').

```
#define  HUFFMAN_TABLE_SIZE  256

typedef struct _t_node {
   unsigned char terminal;
   unsigned char data;
   short left_node_index;
   short right_node_index;
} t_node;

typedef struct _t_file_node {
   unsigned char data;
   unsigned short left_node_offset;
   unsigned short right_node_offset;
} t_file_node;
```

```
t_node list[HUFFMAN_TABLE_SIZE];
short list_size = 0;
short root_node_index = -1;
```

Como se puede ver reservamos espacio para guardar un árbol Huffman de 256 nodos. Dependiendo de la aplicación deberemos ajustar el tamaño del array para aprovechar al máximo la escasa memoria del MSX. Para cargar el árbol Huffman tenemos el siguiente código:

```
/* read_node es una función recursiva que va
   leyendo los nodos de fichero */
short read_node(char fd, unsigned short offset,
short *next_index, t_node *list) {
   t_file_node file_node;
   t_node node;
   short ret_index;

   /* buscamos el nodo en el fichero y lo
      cargamos */
   lseek(fd, offset, SEEK_SET);
   read(fd, &file_node, sizeof(t_file_node));
   /* obtenemos de forma recursiva el nodo hijo
      izquierdo, si procede */
   if (file_node.left_node_offset > 0)
       node.left_node_index = read_node(fd,
file_node.left_node_offset, next_index, list);
   else
       node.left_node_index = -1;
   /* obtenemos de forma recursiva el nodo hijo
      derecho, si procede */
   if (file_node.right_node_offset > 0)
       node.right_node_index = read_node(fd,
file_node.right_node_offset, next_index, list);
   else
       node.right_node_index = -1;
   /* marcamos, en memoria si somos un nodo
      terminal o no */
 if ((node.left_node_index != -1) ||
(node.right_node_index != -1)) {
       node.terminal = 0;
       node.data = 0;
   }
   else {
       node.terminal = 1;
       node.data = file_node.data;
   }
   /* copiamos el contenido del nodo en la posi-
      ción apropiada del array de nodos */
   memcpy(&(list[*next_index]), &node,
sizeof(t_node));
   ret_index = *next_index;
   printf("index = %d\n\r", ret_index);
```

```
    (*next_index)++;
    /* devolvemos el índice dentro del array de
      nodos en el que está el nodo que acabamos
      de cargar */
    return ret_index;
}

/* lee todo el árbol huffman de fichero */
short read_tree(char fd, t_node *list, short
*list_size) {
    unsigned short root_node_offset;
    short next_index;

    /* obtenemos el offset del nodo raiz */
    lseek(fd, 3, SEEK_SET);
    read(fd, &root_node_offset, sizeof(unsigned
short));
    /* inicializamos next_index y cargamos de
     forma recursiva todos los nodos del árbol */
    next_index = 0;
    root_node_index = read_node(fd,
root_node_offset, &next_index, list);
    *list_size = next_index;
    /* ya tenemos el árbol cargado en memoria */
    return root_node_index;
}
```

Una vez tenemos el árbol cargado en memoria ya podemos proceder a ir al área de datos del fichero e ir descomprimiendo:

```
void uncompress(char input_fd, t_node *list,
short root_node_index, char output_fd) {
    unsigned short byte_offset;
    unsigned short bit_offset;
    unsigned char current_byte;
    unsigned char current_bit;
    t_node *current_node;
    short readed;
    long compressed_size_bits, offset_bits;

    /* obtenemos el offset de los datos comprimi-
       dos */
    lseek(input_fd, 5, SEEK_SET);
    read(input_fd, &byte_offset, 2);
    /* obtenemos el tamaño en bits de los datos
       comprimidos */
    lseek(input_fd, byte_offset, SEEK_SET);
    read(input_fd, &compressed_size_bits,
sizeof(long));
    printf("compressed_size_bits = %ld (at
%04x)\n\r", compressed_size_bits, byte_offset);
    /* leemos el primer byte */
   printf("data starts at %04x\n\r",
lseek(input_fd, 0, SEEK_CUR));
    readed = read(input_fd, &current_byte, 1);
    /* inicializamos algunas variables auxiliares
    */
    bit_offset = 0;
    offset_bits = 0;
    /* empezamos en el nodo raiz */
    current_node = &list[root_node_index];
    while (readed == 1) {
        if (bit_offset > 7) {
            bit_offset = 0;
            byte_offset++;
            readed = read(input_fd, &current_byte,
1);
        }
        current_bit = (current_byte >>
bit_offset) & 1;
        if (current_bit) {
            /* tenemos un 1 en la entrada: nos
            vamos al hijo derecho en el árbol */
            current_node = &list[current_node-
>right_node_index];
        }
        else {
            /* tenemos un 0 en la entrada: nos
            vamos al hijo izquierdo en el árbol */
            current_node = &list[current_node-
>left_node_index];
        }
        if (current_node->terminal) {
           /* si es un nodo terminal, emitimos el
campo dato y regresamos al nodo raiz del árbol */
            unsigned char byte_data = (unsigned
char) current_node->data;
            write(output_fd, &byte_data, 1);
            current_node = &list[root_node_index];
        }
        bit_offset++;
        offset_bits++;
        /* hasta que ya no haya más bits que
        procesar */
        if (offset_bits >= compressed_size_bits)
            break;
    }
}
```

De esta manera escribiremos en el fichero de salida los datos sin comprimir. En este ejemplo se ha hecho una descompresión desde fichero a fichero, pero nada impide hacerla desde fichero a memoria o incluso desde memoria a memoria. La he escrito así para que sea más pedagógica. En la función main lo único que hacemos es llamar primero a leer el árbol y luego a descomprimir:

```
int main(void) {
    char input_fd;
    char output_fd;

    input_fd = open("FROM.HUF", O_RDONLY);
    if (input_fd < 0) {
        printf("error opening compressed input
file FROM.HUF\n");
        return 1;
    }
    output_fd = creat("TO.TXT", O_WRONLY,
ATTR_ARCHIV);
    if (output_fd < 0) {
        printf("error opening output file
TO.TXT\n");
        return 1;
    }
    root_node_index = read_tree(input_fd, list,
&list_size);
    uncompress(input_fd, list, root_node_index,
output_fd);

    close(input_fd);
    close(output_fd);
    return 0;
}
```

A modo de ejemplo, en el fichero huffman.tar.gz hay un texto con la obra "Veinte Poemas de Amor y una Canción Desesperada" del poeta Pablo Neruda que, en formato texto normal ocupa 21156 bytes, y que, comprimido, ocupa 12560 bytes.

ALGUNOS DATOS DE INTERÉS

A la hora de comprimir texto el algoritmo Huffman logra aproximadamente entre un 40 y un 60% de ratio de compresión, mientras que para ficheros binarios e imágenes planas el ratio de compresión desciende a un 20 o 30%. Existen, por supuesto, otros algoritmos más elaborados. El que implementa el PuCrunch consigue ratios mejores: De un 80% para ficheros com muy poca entropía (i.e., imágenes con muchos colores planos) y en torno a un 40 a 60% por término medio en ficheros de imágenes y código. A continuación he puesto una pequeña tabla con la comparativa de ratios de compresión entre el algoritmo PuCrunch y el algoritmo de Huffman, los ficheros de ejemplo son los utilizados por el PuCrunch para realizar sus propias comparativas entre otros algoritmos: "delenn.bin" es un visualizador con una imagen com mucho dithering, "sheridan.bin" es un visualizador con una imagen con algunas áreas negras y "ivanova.bin" es un visualizador con una imagen con grandes áreas negras. Todos los ficheros se pueden descargar de:

http://www.cs.tut.fi/~albert/Dev/pucrunch/#Res1.

Ficheros de ejemplo	PuCrunch	BitBuster	Huffman
delenn.bin	58.2%	55.6%	42.7%
sheridan.bin	73.4%	72.9%	61.4%
ivanova.bin	79.2%	78.6%	66.6%

El ratio de compresión se ha calculado de la siguiente manera: (1 - (TamañoComprimido / TamañoOriginal)) * 100 (en porcentaje). De tal forma que si el tamaño original es igual al tamaño comprimido el ratio es del 0% mientras que si el tamaño comprimido es 0, el ratio de compresión sería del 100%. Un fichero comprimido con la mitad de tamaño que el fichero original implica un ratio de compresión del 50%.

Como se puede ver, la capacidad de compresión de los dos algoritmos ya existentes y avanzados como el PuCrunch y el BitBuster supera a la del algoritmo de Huffman puro. El algoritmo de Huffman suele utilizarse como algoritmo auxiliar o base para otros algoritmos más elaborados y que cogen prestadas ideas de otros algoritmos como el RLE o el LZW (tal es el caso de la compresión ZIP que utiliza una combinación de LZW y Huffman). El gran problema que posee el algoritmo Huffman, que es el tamaño de su árbol de codificación, puede ser solventado, en parte, comprimiendo con otro algoritmo este árbol (con LZW, LZ77 o RLE por ejemplo).

CONCLUSIÓN

Espero que este artículo haya servido para despertar el interés por este tipo de algoritmos en el lector. La compresión Huffman es uno de los métodos clásicos para comprimir datos aunque no es ni de lejos "el mejor algoritmo de compresión", da buenos resultados comprimiendo texto aunque no tan buenos comprimiendo imágenes o datos binarios puros. Nótese que, para el caso de que queramos comprimir una imagen de 16 colores, el alfabeto de partida no tendrá 256 valores posibles, sino sólo 16 con lo que obtendremos, por lo general, árboles Huffman más compactos. Por otro lado no tenemos porqué generar un árbol Huffman para cada imagen o texto, sino que podríamos concatenarlos todos y generar un único árbol Huffman que pudiese descomprimirlos todos. Esta última solución sería un poco menos eficiente desde el punto de vista teórico, pero en la práctica supone no tener que guardar muchos árboles Huffman en disco.

Todo el código fuente relacionado con este y otros artículos que he publicado en esta revista se puede bajar de http://msx.gabiot.com, mi página sobre MSX, y es GPL. Cualquier duda, sugerencia o corrección será bienvenida a avelinoherrera@hotpop.com.

MAGNUM (EMPIRE CITY 1931)

Existen juegos poco conocidos pero que merece la pena 'descubrir'. Más allá del horizonte "rc" podemos encontrar software digno de mención. Es el caso del megarom de primera generación que ahora nos ocupa...

ETIQUETAS

Podríamos comenzar hablando del género al que pertenece: para clasificar el juego sin meter mucho la pata echaré mano de referencias.

Todos conocemos, sin duda, el clásico "OPERATION WOLF" de Taito (aunque sea por la insufrible conversión pseudomsx1, que en realidad es una pésimo port de la MUY digna versión speccy).



En pocas palabras consiste en abatir a un cierto número de enemigos de diferente tipo y envergadura mientras el scroll horizontal se mueve a velocidad fija, manejando un puntero que marca el lugar de impacto de las diferentes armas con que contamos.

Por otro lado, en el ámbito MSX disponemos de "COLT 36", un divertido título con el que Topo nos sorprendiera junto a otro par ("Temptations" y la conversión oficiosa del "METROCROSS" de Namco, "Ale Hop!").



En Colt36 también abatimos a diferentes enemigos y objetos, pero en esta ocasión tenemos el control del scroll (vertical), mediante movimiento del puntero de impacto.

El ambiente es mucho más divertido, con un toque infantil y festivo. Los enemigos, 1 a 1 (frente a las multitudes de OpWolf).

Pues bien, el megarom del que hablamos estaría a medio camino de ambos. Un poquito más cerca de Colt36, pero "fundiendo" características de ambos:

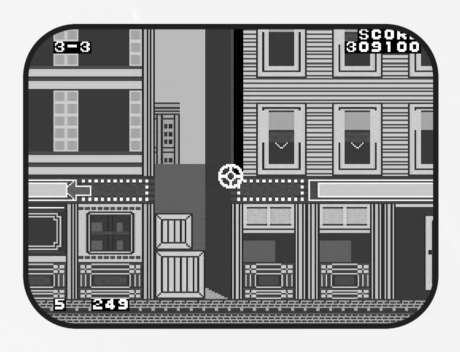
- Scroll horizontal, pero controlable por los movimientos del puntero que maneja el jugador.

- Ambiente más bien serio, pero no exento de un toque 'irreal' o peliculero que sin duda suaviza el hecho de que al fin y al cabo el juego consiste en abatir enemigos a golpe de metralleta.

- Y en cuanto a dificultad, de nuevo término medio: ni tan difícil (o arcadiano) como OpWolf ni tan sencillo (o limitado) como Colt36.

Queda pues descrito el tipo de juego. Se aceptan propuestas oficiales de "etiquetación". En principio pensé en usar algo como "shooter en perspectiva personal con scroll horizontal 2D". Pero no me convencía. Me parece que lo de 'perspectiva personal' se acerca más a títulos como "Doom" o "Counter Strike". Si os gusta ese género os recomiendo que consigáis un 'gunstick' (msx-compatible!) y disfrutéis del megarom "DUNGEON HUNTER" (Ascii, 1989). Otro título poco conocido que tampoco os defraudará (muy cuidado a nivel estético y sonoro). Salvando las evidentes diferencias técnicas con respecto a los citados títulos de PC, la acción y diversión están igualmente aseguradas.

Pero volvamos al juego que nos ocupa...



VERSIONES

El origen de todo es la recreativa "Empire City: 1931", que llegó a los salones de la mano de Seibu Kaihatsu (Taito poseía la licencia para el territorio japonés) en el año 86. Un año más tarde la casa Toshiba Emi/I.S.I. se ocupó de portar el arcade a 2 máquinas domésticas: NES y MSX. La única pérdida apreciable (obviando los evidentes recortes visuales) era el scroll vertical... y con él buena parte de la alta dificultad original. En ambos casos, NES y MSX, mismo título: "Magunam Kiki Ippatsu, Enpaia Siti 1931" (llamémosle MAGNUM para abreviar). Las 2 versiones parecen estar cortadas por el mismo patrón, pero sin que ninguna de ellas esté basada en la otra de un modo claro. Es decir, planificación común pero sacando partido de cada máquina concreta. Al menos aparentemente...

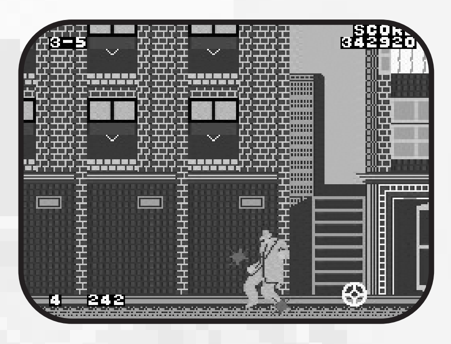


Esto denota, para empezar, un cierto cuidado y buenhacer. Aunque no me importaría que la versión MSX estuviese basada en la de NES, puesto que en general los juegos que provienen de la pequeña de Nintendo suelen dar buen resultado en MSX (Higemaru, Dragon SlayerIV ... bien). Sí sería más preocupante, en cambio, que el juego hubiese llegado a MSX desde Sega Master System (Rastan Saga, Heroes Of The Lance ... mal).

EL JUEGO

Cual un Elliot Ness al uso, nos ponemos en la piel de un policía para luchar contra los gángsters en los años '30.

Controlamos un puntero móvil que movemos con libertad y con el que podemos provocar scroll horizontal al 'empujar' los extremos laterales. Una flecha azul aparecerá en los laterales de la pantalla para indicarnos la dirección en la que aparecerá el próximo enemigo a batir: a pie de calle, en las ventanas de los edificios, saliendo de las alcantarillas, en el campanario de la iglesia, escondidos tras la puerta de un coche... en cualquier parte puede aparecer un maloso dispuesto a dispararte si tú no lo haces antes.



El juego se estructura en ROUND-STAGE. Cada round (o 'vuelta') está compuesta por 8 stages (o 'fases'). 8 estupendos entornos gráficos diferentes, pues, siendo el octavo el lugar donde encontramos al "boss". Para pasar de un entorno (fase) al siguiente basta con eliminar a un número determinado de enemigos. Vencido el boss de cada fase8, se comienza una nueva vuelta elevándose la dificultad general y encontrando a un nuevo boss al final de la vuelta.

En la primera vuelta el número de enemigos a abatir por fase es muy bajo. En cada nueva vuelta se eleva dicho número, a la par que el contador de tiempo permanece igual... elevándose de este modo la dificultad. También, a lo largo del juego, van apareciendo enemigos con nuevos comportamientos y/o emplazamientos.

De modo que, en definitiva, se consigue un primer contacto fácil y ameno, de aprendizaje, para luego aumentar progresiva e inteligentemente la dificultad y así mantener el interés del jugador y obligarle a mejorar.

Es importante entender y dominar el sistema de disparo: ráfagas de 10 impactos con, ojo, un tiempo de recarga entre ellas. Esto quiere decir que desde que se dispara la décima bala de una ráfaga y hasta poder volver a disparar de nuevo pasa un tiempo en el cual no se puede disparar. Un corto periodo de cierta (pero NO completa) indefensión.

Al principio del juego, en la primera vuelta, aconsejo usar una ráfaga completa contra cada enemigo. Son pocos enemigos al principio, como ya he dicho... y es aconsejable habituarse al control del puntero y aprender cómo acertar. Además de que con cada bala que acierta se incrementa la puntuación y podemos conseguir vidas extra al alcanzar ciertas cantidades. Aprovechad pues las primeras fases para engrosar el marcador descargando el cargador en cada enemigo. Más adelante será complicado.



Conforme avanzamos en el juego se impone el hacer un uso de semi-ráfagas (de 5-5 ó 3-3-4) para economizar balas... o sea, tiempo. Marcador de tiempo y balas están relacionados en cierto modo, como explicaré a continuación. En ocasiones se hará necesario disparar lo justo, 1 a 1, cuando el marcador de tiempo se acerca peligrosamente a 0 (o a partir de la cuarta vuelta aprox). En cualquier caso, siempre procurando llevar la cuenta del cargador de balas para no enfrentarse a un enemigo justo en el delicado momento de recarga.



El marcador de tiempo comienza siempre en 250 y decrece a ritmo constante, salvo cuando disparamos... momento en el cual decrece al ritmo de disparo. De ahí la conexión entre tiempo y disparos. Una ráfaga completa de 10 disparos seguidos es un lujo que no siempre se puede uno permitir.

Otra característica importante a tener en cuenta es el segundo botón de disparo, que nos permite CUBRIRNOS contra un disparo enemigo durante un instante. Hay que ajustar muy bien el momento de pulsar, pero este botón puede y suele ser de bastante utilidad. Junto a un uso inteligente y medido del sistema de "ráfagas de 10", el cubrirse es básico en ciertos momentos del juego (sobre todo en ciertos enemigos móviles y en bosses).

Al igual que una flecha azul nos indica la situación del próximo enemigo a abatir, puede aparecer en ocasiones, en la esquina inferior izquierda de la pantalla, una nube (o "bocadillo") con un contador de tiempo desdendente... 5 a 0. Esto quiere decir que vamos a recibir un disparo de modo inminente. Momento, claro, de prestar especial atención al segundo botón de disparo.

Como curiosidad, cabe resaltar que en MSX se ha usado el modo de sprites ampliados para ilustrar este movimiento. Lo cual se agradece sin duda pues, sin llegar al gran tamaño del gráfico original de la recreativa... sí que se consigue un mayor tamaño (y mejor efecto) que en la versión de NES.

Además de gángsters de todo tipo, también aparecerán pequeñas cajas al nivel del suelo. Conviene pararse un instante y dispararles, pues suelen contener items interesantes (como balas extra). Bolsas de dinero o lingotes que pueden ser disparados varias veces para obtener puntuación ascendente... e incluso pajáros o bichos saldrán de la caja al dispararle. Puede ser divertido entretenerse con las cajas, al menos al comienzo del juego (cuando aún es fácil y sobra tiempo). Además de por los items que liberan, las cajas son útiles para practicar puntería o comprobar, si se ha perdido la cuenta, el estado del cargador (terminar de descargar una ráfaga y esperar el tiempo de recarga antes de salir en busca del siguiente enemigo).

Y, ojo, sólo contienen items las pequeñas cajas de color amarillo. Es fácil despistarse con otras cajas, sobre todo en la fase de la fábrica (donde hay unas cajas pequeñas, aunque de color gris).

ASPECTOS TÉCNICOS Y CONCLUSIONES

El aspecto gráfico está bastante logrado. El diseño de patrones es inteligente y el uso del color es curioso e interesante.

Junto al apartado sonoro, puede hablarse en general de una GRAN AMBIENTACIÓN. Está todo bastante pensado y conseguido. Destacable sin duda el logro estético y sonoro.

Existen otros juegos de temática similar (CosaNostra, Chicago's 30, The Untou-chables...). Pero además de por el tipo y sistema de juego, este megarom destaca sobre todos esos juegos por la notable ambientación. Y aunque esta destacable virtud pudiera en principio deberse tan sólo a la recreativa original, no sería justo pasar por alto el buen trabajo que se ha hecho en las máquinas receptoras del port: tanto NES como MSX son, sobra decirlo, inferiores en cuanto a hard con respecto a la recreativa original... y aún así se ha capturado el espíritu original de un modo envidiable.

Obviamente, el "scroll" no es ni suave ni perfecto. Es quizá el punto más flaco (o el más visible/apreciable de los defectos) en el juego. Pero tampoco afecta demasiado al conjunto. Y sea como fuere, se agradece el valor de arriesgarse por parte de Toshiba Emi (o de otro modo ahora no podríamos disfrutar de este juego, como de hecho ha ocurrido con tantos títulos basados en scroll horizontal).

La jugabilidad es buena. Como cualquier otro juego, puede conllevar un cierto grado de aprendizaje (de control) al principio. Esto puede depender también de cada jugador concreto, de su habilidad personal e interés por el juego y/o por el género.

La dificultad, como ya he dicho, está bien ajustada. El juego es equilibrado y adictivo: un comienzo accesible y entretenido, un reto creciente conforme se avanza.

En definitiva, un título recomendable y no exento de una cierta sana ambición... con resultados como mínimo destacables. En un género poco común y muy olvidado.

Técnicamente deseo insistir en el uso de 'pseudoscroll' a patronazos. Es algo que quizá asustaba por varios y diferentes motivos... muchos de ellos plenamente justificados (hay que admitirlo!). Pero si se hubiera usado más esta arriesgada feature, existiría sin duda una mayor variedad en la juegoteca MSX (os invito a revisar los títulos de ZAP creados para Sony en torno a 1984: son todo un alarde de color y valentía que personalmente valoro cada vez más).

Objetivamente, y dejando de lado el resto de consideraciones, estamos ante un juego que llena un vacío de género bastante evidente. Sólo por ello, fans de OpWolf y Colt36, debéis probarlo.

Así que este 'MAGNUM' se merece, por todo lo expuesto hasta este punto, un vistazo y unas partidas. ¡Qué menos!

Y es que, POR FORTUNA, no todo son konamis. Pues ni los konamis eran siempre mejores ni los demás eran siempre peores. Ya va siendo hora de adquirir un poquito de perspectiva, no?

Sut (sutchan@wanadoo.es)

Sprites en Screen2

PROLOGO

Este es el típico artículo que se suele escribir en las revistas de MSX desde el principio de los tiempos, y el cual el 98% de los lectores pasarán de largo sin contemplaciones. Un artículo de programación, y encima en ensamblador. Dos temas tabú para toda la gente que solo quiere disfrutar de la faceta lúdica que nos brinda el sistema.

Pero, seguro, que este 98% de lectores no se imagina, que gracias a este tipo de artículos, escritos en otra época por gente de la que hoy se denomina como "frikis", mucha gente se labró su futura profesión y pasión, y gracias a ella, muchos hemos podido disfrutar de grandes momentos con ella, y lo que es mejor, gracias a la cual todos hemos disfrutado de grandes juegos en este extraño y variopinto sistema denominado MSX.

Así pues, antes de cometer la crueldad de saltar estas páginas, de la que en estos momentos es la UNICA REVISTA EN PAPEL que habla de MSX, haz un esfuerzo por intentar leer y comprender los párrafos que a continuación tienes impresos. Si al finalizar la lectura, te has quedado igual, o mejor dicho, sigues sin querer saber nada del tema, habrás aprovechado estos minutos para algo interesante.

INTRODUCCION

Debido a la proliferación en los últimos años de nuevos programas para los MSX de primera generación, en su gran mayoría, videojuegos, mucha gente se ha visto interesada en añadir su granito de arena en esta empresa. La gran mayoría de esta gente, ha visto como, con un pequeño esfuerzo y gracias al potente BASIC que incorpora el MSX, conseguía en poco tiempo dar vida a su programa soñado. Pero, también, en ese mismo tiempo, pequeño, se habrá dado cuenta que si quiere que realmente su sueño cobre vida, tal como lo tenía pensado, con BASIC lo tiene difícil, y que lo que esperaba ver al ejecutar el programa y lo que realmente ha visto al ejecutarlo, no se parecen en nada. O no, puede que los resultados sean satisfactorios, o se piense que los resultados son satisfactorios. Si te encuentras en ese grupo de gente, puedes pasar estas páginas. Pero si por el contrario, tu nivel de superación y autoexigencia es grande, y deseas crear realmente ese programa soñado, sigue leyendo atentamente.

En este pequeño artículo, vamos a intentar explicar cómo posicionar un sencillo sprite en pantalla en Screen 2, con modo 1 de sprites, y sprites de 16x16. Es decir, posicionar un sprite en MSX1 como el 99% de los programas para MSX1 existentes. No es intención del artículo explicar cómo funcionan los sprites en MSX, ni cómo funciona el MSX, ni dar clases de ensamblador. Sobre estos particulares existen cientos y cientos de páginas de información y pueden ser consultados en cualquier momento. Aquí iremos al grano, y lo intentaremos hacer de una forma sencilla. Lo que buscamos con este pequeño artículo es dar una pequeña base inicial para cualquier programa.

También quiero aprovechar para dar a conocer una gran herramienta que nos brindan la mayoría de los ensambladores del sistema, sean nativos o sean cross-assemblers, y es el preprocesado. Esto nos puede ayudar mucho, sobre todo para todos aquellos que utilicen el propio MSX como plataforma de desarrollo.

PREPROCESADO

El preprocesado, es una herramienta de la cual disponen muchos ensambladores, a través de la cual podemos generar diferentes programas finales, sencillamente cambiando el valor de unas variables. Quizá al lector más joven o familiarizado con el C, le suene más herramientas como el CPP, que no es ni más ni menos que un programa que realiza las funciones de preprocesado del fichero que pasemos como parámetro, generando un fuente final dependiente de esas variables de preprocesado. Un galimatías es lo que estoy montando. Mejor expliquemos esto con un ejemplo.

Nuestra intención es generar un programa que funcione tanto en MSXDOS, como en una ROM. No es algo habitual ni común, pero puede ser muy útil si programamos directamente en MSX. Ejecutar una ROM como tal, exige, por norma general, reiniciar el sistema una vez ejecutada, o bien tener muy bien preparada la salida del mismo para que nos retorne al MSXDOS, donde tenemos nuestras herramientas de desarrollo, tales como el editor de texto, el ensamblador, etc, etc ... Por ello es más cómodo hacer un programa para MSXDOS, pues podemos volver al mismo una vez ejecutado nuestro programa, para seguir trabajando en el mismo.

Lo intentaremos con un ejemplo más simple. Mientras

desarrollamos nuestro juego, necesitaremos, tener trucos activados, tales como vidas infinitas, inmunidad, poder acceder a algún menú secreto, empezar desde cualquier pantalla... Pero queremos a su vez, que cambiando una simple definición, todo se ensamble para una versión final, donde desaparezca todo ese código de "cheat". Una forma, la clásica de hacer esto es repasar el programa, quitar ese código e inicializar esas variables con los valores iniciales. Pero, ¿qué pasa si necesitamos añadir más código? Pues que no tenemos más remedio, que volver a colocar esos trucos. Y así podemos repetir la operación una y otra vez, con el consecuente peligro de cometer un olvido en la última compilación, dejando algo en el programa que no deseábamos, provocando que nuestro programa salga mal parado.

¿Y cómo conseguimos eso de una forma fácil? Pues aquí es donde entra el preprocesado, gracias al cual se ensamblarán unas partes u otras del programa, dependiendo de unas pequeñas definiciones, y consiguiendo así no tener código ensamblado que no utilicemos y que nos ocupe un precioso espacio en nuestro programa.

NESTORMODE

Todos nos hemos reído escuchando y leyendo artículos de programación Néstor, y todos tendremos retenidas sus palabras más habituales, aquellas que nos repite una y otra vez, de respetar el estándar, programar con clase, seguir las normas, etc, etc ...

Si bien, para programas de utilidad, parece ser que estas normas son muy respetadas, en el sector lúdico se suelen ignorar completamente. La mayoría de las veces, y esto es así, por la sencilla razón de obtener velocidad en nuestros programas. Utilizar la BIOS para todo es lento, muy lento, sobre todo para realizar videojuegos. Y esas normas son saltadas en la mayoría de este tipo de programas, sobre todo, en lo que el acceso al VDP se refiere.

Por eso mismo, la normal común es acceder al VDP de forma directa, a través de puertos del Z80. Pero esto, que es mucho más rápido, puede no ser todo lo correcto que imaginamos. Si bien, gracias al MSX2 se estandarizó el uso de puertos de valor fijo (098h y familia), para acceder al VDP, dictados por ASCII, en MSX1 la cosa no es así. Aunque la gran mayoría de los MSX tienen esos valores fijos, y es por eso por lo que hay un gran nivel de compatibilidad accediendo directamente a ellos. Pero la forma correcta de realizarlo es averiguar cuáles son esos puertos, si queremos realmente acceder a ellos directamente.

Otra cosa que no se realiza mucho, y queda realmente bien, y da un toque de calidad a nuestros programas, en formato MSXDOS es retornar al sistema manteniendo los valores que el usuario tiene definidos antes de arrancar nuestro programa, y que suelen ser modificados por norma general.

Por todo esto, y, como homenaje *no* póstumo a Néstor, vamos a dotar a intentar que el programa sea más correcto y sin que por ello se vea mermada la velocidad de ejecución del mismo. Hablar bien, no cuesta una mierda.

EL PROGRAMA

Intentaremos de una forma breve, las partes del programa adjunto. No vamos a explicar exhaustivamente cada parte, pues saldría del propósito del artículo. Si éste tuviese éxito (5 lecturas más o menos) podríamos plantearnos hacer algo que empezase desde un nivel más bajo y con explicaciones más detalladas.

- *ROMMODE* : Este define es para el preprocesador. Gracias a éste, si su valor es 0 (falso), el programa se ensamblará como un archivo COM para MSXDOS. Si su valor es 1 (cierto), el programa se ensamblará como una ROM, de 8k (8192 bytes). Las variaciones son mínimas. Cambios a las llamadas a la Bios, ubicación de las variables e inicialización del programa, así como la dirección de ensamblado.

- *Msx System Vars, Bios calls y Screen 2 values*: Estas son las variables del sistema, llamadas a la bios, y valores de Vram de Screen 2. Para más información, podemos consultar el Technical HandBook, o cualquier documento técnico sobre el sistema MSX.

- *Program_Init*: Aquí es donde el preprocesador empieza su tarea. Un programa ROM necesita de una cabecera para que el sistema lo reconozca como tal al inicializarse y lo ejecute, así como ser ensamblado a partir de la dirección 04000h. En esta pequeña cabecera se especifica la dirección de inicio de programa. Un programa COM no necesita ningún tipo de cabecera, comienza sin más, y se ensambla a partir de la dirección 100h. Dependiendo de la constante ROMMODE obtendremos una u otra inicialización.

- *Initmain*: Una pequeña rutina de inicialización del programa. Dependiendo del valor para preprocesador ROMMODE, o bien inicializaremos el puntero de pila y pondremos al Z80 en IM1, o guardaremos el puntero de pila para salir del programa, en el momento que queramos. En ROMMODE no podremos salir del programa. Por lo demás en esta pequeña inicialiación, llamaremos a las subrutinas que nos inicializará el modo de video y cargará nuestro sprite en la vram. Seguidamente damos un valor inicial a las variables de posicionamiento de nuestro sprite, situándolo en el centro de la pantalla, por últi-

mo activamos la pantalla (que habrá sido desconectada en la rutina init_screen).

También seleccionamos el patrón adecuado para los planos donde posicionaremos nuestro sprite. Cuando utilizamos sprites de 16x16, la Vram puede alojar 64 patrones de sprites definidos simultáneamente. En modo de 8x8, 256. Cada patrón es numerado de 0 a 255 en ese modo. Pero en modo 16x16, para acceder a un patrón tendríamos que usar algo así como:

patronsprite = 4 * n_patron.

Donde n_patron es un número de 0 a 63. En modo 16x16 cada sprite está compuesto por 4 patrones de 8 x 8. Así pues, cuando le decimos al VDP que posicione "x" patrón en "y" plano, en modo 16x16, podremos usar cualquiera de los 4 que lo definen. De esta manera el patrón 0-3 corresponderá con el primer patrón, el patrón 4-7 con el segundo, etc ...

Esta última parte es interesante, y de nuevo nos valemos del preprocesador para preparar el programa para MSXDOS o ROM. La llamada en formato ROM es directa, ya que la BIOS ha de estar conectada en la página 0. En MSXDOS la BIOS no está conectada y para llamar a una rutina situada en otra página y slot, debemos de hacer uso de la rutina del sistema CALSLT, que tenemos disponible en MSXDOS, BASIC o arrancando en ROM. Lo utilizaremos en varias partes del programa.

- *Main*: El bucle principal del programa, la parte más corta. En este bucle, llamaremos en cada interrupción a dos subrutinas. La primera (paintspr) nos actualizará los datos del sprite con los nuevos valores que obtendrá de la siguiente rutina.

Puede esto parecer confuso, pero tiene su explicación, pero no es motivo de este artículo tampoco dar una explicación detallada. Quedémonos con un dato: Los datos gráficos, es decir, todo lo que se pinta en pantalla, han de ser actualizados en la denominada VBL, que es, a grandes rasgos, el momento donde nuestro monitor o televisor no está dibujando la pantalla. De esta manera, evitamos parpadeos, que seguro que alguna vez hemos visto en algún programa. Las interrupciones en MSX son producidas por el VDP, y se nos garantiza que la interrupción principal, se produce durante el tiempo de VBL. Así pues, después de esperar la interrupción (halt), actualizamos esos datos.

La segunda subrutina (movespr), realiza varias funciones. Chequea el teclado, y actualiza los valores de nuestro sprite, dependiendo de esos datos obtenidos. Y por último vuelta a empezar, pues el bucle es infinito.

- *init_screen*: Inicializa Screen 2, activa los sprites de 16x16 modo 1. Si el programa se ensambla en modo DOS, salvamos en unas variables los datos de screen, colores y el valor de la variable de sistema del click de teclado, antes de cambiarlos. De esta manera y sólo en DOSMODE al salir del programa serán restaurados. Seguidamente cambiamos el color del borde y fondo a 0 (negro), y llamamos a la rutina de la BIOS CHGMOD, para posicionar Screen 2. La BIOS es muy cómoda para estas operaciones, que de haberlas hecho mediante escrituras directas en VDP necesitarían de varias escrituras. Siendo operaciones que realizamos pocas veces, lo mejor es utilizar la BIOS para ellas. Seguidamente llamamos a initport, para obtener los valores de los puertos del VDP.

Por último posicionamos los sprites en modo 16x16, a la vez que actualizamos la variable del sistema donde se encuentra la copia del registro del VDP que hemos alterado. Esto es muy recomendable para asegurarnos la compatibilidad con otros programas y con la propia BIOS. Los registros de 0 a 7 del VDP son de sólo escritura. Nunca podemos saber qué valor poseen. Para paliar este problema, la BIOS escribe en una zona de memoria de las variables del sistema, el valor que introduce en el registro. De esta manera puede saber en todo momento como está ese registro. Así que nosotros, hemos de seguir esa pequeña norma.

- *initport*: Obtenemos el valor de los puertos del VDP. Estos valores se encuentran en la dirección 7 de la BIOS ROM. El valor del port#0 para ser exactos. El puerto#1, es obtenido sumándole uno a ese valor, según nos dictan las normas del estándar. Aquí vemos dos maneras de obtener esos valores. En ROMMODE accederemos directamente a la dirección para obtener el valor. En DOSMODE lo haremos a través de la rutina RDSLT, que nos lee el valor de una dirección de un slot y página dada.

- *initvram*: Un ejemplo de cómo actualizar el puntero de la Vram a una dirección pasada en el registro HL. Esta rutina sólo es válida para posicionar la Vram a valores entre 0-03FFFh, es decir un rango de 16k, que es con lo que cuentan todos los MSX1 de VRAM. Para acceder a posiciones más altas, en un MSX2 o superior, esta rutina no es válida, además de que no funcionaría bien, si pasásemos valores más altos de 03FFFh en HL. La rutina está preparada para posicionar el puntero para escribir o para leer de VRAM. Un dato a tener en cuenta, el NOP que se ejecuta entre las dos instrucciones OUT al port#0 del VDP. Es una medida preventiva para garantizar que el VDP ha procesado el comando anterior. El VDP es lento, y necesita una espera de unos milisegundos para atender al siguiente comando. Si mientras se está procesando un dato, se ejecuta otro comando al VDP, corremos el riesgo de que no se ejecute, con consecuencias para nosotros desastrosas. Esto ocurre con toda seguridad si nuestro Z80 va a más de 3,5 Mhz (y con esta velocidad también es posible). Así que para evitar el problema en los Turbo R, el propio hard

del Turbo R añade esa pausa automáticamente.

En los ordenadores MSX que dispongan de un Z80 a más velocidad, tendríamos problemas. Habría que adaptar ese retardo (el nop) a la velocidad del micro, siendo mayor el retardo a más velocidad. El nop, es funcional en un MSX a 3,5 Mhz.

- *paintspr*: Una de las dos rutinas principales del bucle principal del juego. Escribe los 4 bytes de datos de nuestro sprite, en la dirección del plano 0 de sprites (recordemos que son 32 los disponibles), actualizando su posición o color en caso de haber cambiado.

Remarcar un detalle. La Y de un sprite en MSX no corresponde con la Y gráfica. La coordenada Y 0 del sprite, corresponde a la coordenada 1 de la pantalla gráfica. ¿Por qué? Habría que hablar con los creadores de los chips VDP. Se me ocurren varias teorías, pero no son motivo de este artículo. Aquí, en nuestro programa actuamos con esa coordenada directamente, y no hacemos la corrección. Lo suyo es corregir esa coordenada antes de enviar los datos al VDP restándole uno.

Expliquémoslo un poco mejor. Cuando uno realiza un programa en el que utiliza sprites, normalmente ha de colisionar con otros objetos no sprite que tenemos en pantalla, o bien, es necesario conocer si el sprite se encuentra en una determinada posición para ejecutar un evento adecuado. Para que todo esto nos cuadre, lo mejor es llevar las coordenadas de nuestros sprites de forma natural, sin realizar esa corrección, y sólo cuando vamos a mandar los datos al VDP hacer esa resta de 1, y por supuesto, no modificar nuestra variable. Si todavía no has entendido esto, no te preocupes, lo entenderás.

- *loadsprite*: Cargamos los datos de nuestro sprite, en la vram, en la posición del patrón 0. Con sprites de 16x16, la VRAM dispone de 64 patrones diferentes de sprites cargados simultáneamente.

- *movespr*: La rutina que controla el teclado. Para ello, echamos mano de la rutina BIOS SNSMAT, que nos devuelve el estado de la matriz del teclado. Dado que las teclas que vamos a utilizar son comunes en todos los teclados existentes en el sistema MSX, podemos hacer uso de esta rutina de la BIOS. Y dependiendo del estado de esas teclas escaneadas, actualizaremos la posición de nuestro sprite. Si nos fijamos, son siempre escaneadas y procesadas todas las teclas del cursor, para poder obtener movimientos en diagonal del mismo, a la vez que la tecla SPACE para poder ir cambiando el color del sprite mientras nos movemos. Le hemos puesto al sprite "topes" en los márgenes finales de la pantalla, para evitar que salga de ella.

Estas rutinas de modificación, en caso de usar un sprite DOBLE, actuarán sobre las coordenadas de nuestro segundo sprite.

Finalmente, y sólo en DOSMODE escaneamos el estado de la tecla ESC. Si es pulsada, el programa llamará a la subrutina "exit", para salir al MSXDOS.

- *bioscall*: Esta rutina sólo será ensamblada en DOSMODE. Hace uso de la rutina del sistema CALSLT para llamar a una rutina de la BIOS pasada en IX.

- *exit*: Otra de las rutinas que sólo se ensamblará en DOSMODE.

- *sprdata*: Datos de nuestro sprite. Sprite de 16x16, 32 bytes de datos para definirlo. Remarcar que en la VRAM el sprite de 16 x 16 está almacenado en 4 partes de 8 x 8, y de una forma especial, siendo la primera la parte que corresponde a la zona superior izquierda del mismo, la segunda la parte inferior izquierda, la tercera la superior derecha y la cuarta la inferior derecha.

- *fillff*: Este pequeño código aquí posicionado, es para el ensamblador. En modo ROM, rellenará de datos a 0FFh, hasta 8192 bytes.

- *MEMINIT*: Otra variable para el preprocesador. En caso de ser ROM lo que queremos obtener, posicionaremos las variables en la zona de memoria a partir de 0E000h, para total compatibilidad con un MSX de 8k de Ram. La gran mayoría de juegos en formato ROM de MSX1 lo hacen así. Si el programa se ensambla como un archivo ejecutable de MSXDOS, un COM, las variables estarán posicionadas a continuación de nuestro programa.

NOTAS FINALES

El programa tal cual ha sido testeado en varios ensambladores, tanto nativos como cruzados. Ensambla correctamente con los ensambladores cruzados SjASM, Pasmo y TASM. Y con los ensambladores nativos en MSX, Gen80 y Compass.

Y esto es todo. Os invito a teclear el programa y ejecutarlo en sus distintos modos (ROMMODE o DOSMODE, SINGLESPRITE o DOUBLESPRITE). Veremos un conocido personaje moverse por pantalla. ;)

Puedes bajarte el fichero Test.rom o Test.com en la página de Call MSX: **http://callmsx.gabiot.com**

Armando Pérez Abad

```
; ----------------------
; TEST01.Z8A
; Ejemplo de posicionado
; de un sprite en MSX1
; (c) 2006 Ramones
; -----------------------

; *** Preprocessor defines ***

ROMMODE:                      equ     1              ; 0 Dos Mode 1 Rom Mode

                              if      ROMMODE        ; Dependiendo de ROMMODE, DOSMODE sera
                                                     ; cierto o falso
DOSMODE:                      equ     0
                              else
DOSMODE:                      equ     1
                              endif

SINGLESPRITE:                 equ     0              ; Sprite Sencillo o sprite doble.

                              if      SINGLESPRITE
DOUBLESPRITE:                 equ     0
                              else
DOUBLESPRITE:                 equ     1
                              endif


; *** MSX SYSTEM VARS & BIOS CALLS ***

RGSAV:                        equ     0F3DFh
SNSMAT:                       equ     0141h
HKEYI:                        equ     0FD9Ah
HTIMI:                        equ     0FD9FH
JIFFY:                        equ     0FC9Eh
DISSCR:                       equ     041h
ENASCR:                       equ     044h
CLIKSW:                       equ     0F3DBh
STATFL:                       equ     0F3E7h
EXPTBL:                       equ     0FCC1h
CALSLT:                       equ     01Ch
CHGMOD:                       equ     05Fh
RDSLT:                        equ     0Ch
FORCLR:                       equ     0F3E9h
BAKCLR:                       equ     0F3EAh
SCRMOD:                       equ     0FCAFh


; *** SCREEN 2 VALUES ***

scrpatron:                    equ     0
scrcolor:                     equ     02000h
scrattrib:                    equ     01800h
sprpatron:                    equ     03800h
sprattrib:                    equ     01B00h
vramempty:                    equ     01B80h



; ------------------
; PROGRAM_INIT
; Inicializacisn
; del programa en si
; ------------------

program_init:
                              if      ROMMODE

                              ; *** ROM HEADER ***

                              org     04000h                ; Origen de Rom Standard

                              db      041h,042h             ; Cabecera
                              dw      initmain              ; Inicio de programa
                              ds      12                    ; De momento a 0
```

```
                            else

                            org     0100h               ; Origen en un programa MSX-DOS

                            endif


; -------------------
; INITMAIN
; Inicio del programa
; -------------------

initmain:
                            ; **** ROM CLASSIC INIT ***
                            ; **** NO DISK ****

                            if      ROMMODE
                            di                          ; Inicializacion de una Rom
                            im      1
                            ld      sp,0F380h
                            else

                            ld      (savesp),sp         ; Guardamos la pila para retornar al
                                                        ; DOS en cualquier momento
                            endif

                            call    init_screen         ; Inicializamos Screen 2 y sprites
                                                        ; 16x16

                            call    loadsprite          ; Cargamos en Vram los datos del
                                                        ; sprite


                            ; Inicializacion de variables

                            ld      a,256/2 - 8
                            ld      (Xspr),a
                            ld      a,192/2 - 8
                            ld      (Yspr),a            ; Sprite al centro de la pantalla
                            xor     a
                            ld      (patspr),a          ; Patron de Sprite en Plano 0
                            ld      a,15
                            ld      (colspr),a          ; Color 15 (blanco)

                            if      DOUBLESPRITE

                            ; Inicializacion del segundo sprite, si lo usamos

                            ld      a,256/2 - 8
                            ld      (Xspr2),a
                            ld      a,192/2 - 8
                            ld      (Yspr2),a           ; Sprite al centro de la pantalla
                            ld      a,4
                            ld      (patspr2),a         ; Patron 1 de Sprite en Plano 1
                            ld      a,7
                            ld      (colspr2),a         ; Color 7 (azul)
                            endif



                            if      ROMMODE
                            call    ENASCR              ; Activamos la pantalla
                            else
                            ld      ix,ENASCR
                            call    bioscall
                            endif



                            ; *** MAIN ***
main:
                            ei
                            halt                        ; Esperamos la interrupcion
```

```
                                                                ; (temporizacion)
                              call    paintspr                  ; Escribimos nuevos datos del Sprite
                              call    movespr                   ; Rutinas de movimiento del sprite.
                              jr      main                      ; Repetimos el bucle


; *** INIT ROUTINES ***

; ------------------------
; INIT_SCREEN
; Init Screen 2
; Set Color 0,0,0
; Init 16x16 Mode1 Sprites
; ------------------------

init_screen:
                              if      DOSMODE                   ; Salvamos modo de pantalla y
                                                                ; colores para cuando retornemos al
                                                                ; DOS (solo DOS mode)

                              ld      a,(SCRMOD)
                              ld      (saveinitscreen),a

                              ld      a,(FORCLR)
                              ld      (saveinitcolors),a

                              ld      hl,(BAKCLR)
                              ld      (saveinitcolors+1),hl

                              ld      a,(CLIKSW)
                              ld      (savecliksw),a
                              endif

                              ld      hl,0
                              ld      (BAKCLR),hl
                              xor     a
                              ld      (FORCLR),a                ; Color 0,0,0

                              ld      (CLIKSW),a                ; Quitamos el "click" de las teclas

                              ld      a,2                       ; Screen 2
                              if      ROMMODE
                              call    CHGMOD
                              else
                              ld      ix,CHGMOD
                              call    bioscall
                              endif


                              if      ROMMODE
                              call    DISSCR                    ; Apagamos la pantalla
                              else
                              ld      ix,DISSCR
                              call    bioscall
                              endif


                              call    initport                  ; Buscamos puertos de la Rom

                              ld      a,(vdp_reg99)             ; Sprites 16x16 sin ampliar
                              ld      c,a
                              ld      a,(RGSAV+1)
                              set     1,a
                              ld      (RGSAV+1),a               ; Salvamos el cambio en variable
                                                                ; sistema
                              out     (c),a
                              ld      a,128+1
                              out     (c),a

                              ret
```

```
; -------------------
; INITPORT
; Busca en la Rom
; El valor de los
; puertos del Vdp
; para compatibilidad
; -------------------

initport:


                    if      ROMMODE
                    ld      a,(07h)              ; En posicion 07h de la Rom
                    else                         ; Tenemos el valor de port#0 del VDP
                    ld      a,(EXPTBL)           ; Comunmente 098h.
                    ld      hl,07h
                    call    RDSLT
                    endif

                    ld      (vdp_reg98),a
                    inc     a                    ; Port #1 es Port#0 + 1
                    ld      (vdp_reg99),a
                    ret




; **** VRAM ROUTINES ***

; ---------------------------
; INITVRAM
; Inicializa la VRAM
; A = 1 Escritura
; A = 0 Lectura
; HL : Puntero a la direccion
; ----------------------------

initvram:
                    or      a                    ; Lectura o Escritura?
                    jr      z,initvram0          ; Lectura

                    ld      a,h                  ; Escritura, activamos Bit 6
                    or      64
                    ld      h,a
initvram0:
                    di

                    ld      a,(vdp_reg99)
                    ld      c,a
                    out     (c),l
                    nop                          ; por seguridad para sincronizacion
                                                 ; VDP
                    out     (c),h
                    ret



; -------------------
; PAINTSPR
; Actualiza los datos
; del sprite plano 0
; -------------------

paintspr:
                    ld      hl,sprattrib
                    ld      a,1
                    call    initvram             ; Posicionamos el puntero de la Vram
                                                 ; En la zona de datos del plano 0 de
                                                 ; sprites

                    di
                    ld      hl,Yspr

                    if      SINGLESPRITE
                    ld      b,4
```

```
                            else
                            ld      b,8
                            endif

                            ld      a,(vdp_reg98)
                            ld      c,a
                            otir                        ; Y escribimos sus 4 valores de
                                                        ; datos
                            ret



; -------------------
; LOADSPRITE
; Carga los datos
; graficos de un sprite
; en el patron 0
; -------------------


loadsprite:
                            ld      hl,sprpatron
                            call    initvram            ; Puntero de Vram a patron 0

                            di
                            ld      hl,sprdata
                            if      SINGLESPRITE
                            ld      b,32
                            else
                            ld      b,64
                            endif

                            ld      a,(vdp_reg98)
                            ld      c,a
                            otir                        ; Escribimos los 32 bytes del patron
                            ret




; *** PROGRAM ROUTINES ***

; ------------------------
; MOVESPR
; Chequeo de teclas
; y movimiento del Sprite
; ------------------------

movespr:

                            ld      a,8                 ; Leemos los datos de la 8* fila del
                                                        ; teclado
                            if      ROMMODE             ; que contiene cursores, espacio ...
                            call    SNSMAT              ; directamente en Rom
                            else
                            ld      ix,SNSMAT
                            call    bioscall            ; A traves de Bioscall en DOSMODE
                            endif

                            ld      c,a                 ; Salvamos en C el valor retornado
                                                        ; por SNSMAT pues usaremos
                                                        ; Acumulador en las rutinas de
                                                        ; movimiento.

                            bit     7,c                 ; Cur. Derecho?
                            call    z,movesprright
                            bit     6,c
                            call    z,movesprdown       ; Cur. Abajo?
                            bit     5,c
                            call    z,movesprup         ; Cur. Arriba?
                            bit     4,c
                            call    z,movesprleft       ; Cur. Izquierda?

                            bit     0,c
```

```
                        call    z,movesprcolor          ; Espacio?

                        if      DOSMODE
                        ld      a,7
                        ld      ix,SNSMAT               ; Leemos la fila 7 que contiene la
                                                        ; tecla ESC
                        call    bioscall

                        and     4
                        jp      z,exit                  ; Si esta pulsado volvemos al DOS.
                        endif

                        ret


movesprcolor:           ; SPACE
                        ld      a,(colspr)
                        inc     a
                        and     0Fh                     ; Evitamos que valga mas de 15
                        ld      (colspr),a              ; Vamos cambiando de color el sprite
                                                        ; si esta pulsada la barra
                                                        ; espaciadora

                        if      DOUBLESPRITE
                        ld      a,(colspr2)
                        inc     a
                        and     0Fh
                        ld      (colspr2),a
                        endif
                        ret


movesprleft:            ; C. LEFT
                        ld      a,(Xspr)
                        sub     1                       ; X -=1;
                        ret     c                       ; Cy = 1 Valor menor de 0, no
                                                        ; actualizamos variable
                        ld      (Xspr),a                ; Actualizamos la X
                        if      DOUBLESPRITE
                        ld      (Xspr2),a
                        endif
                        ret
movesprright:           ; C. RIGHT
                        ld      a,(Xspr)
                        inc     a                       ; X+=1;
                        cp      256-16                  ; El valor es mayor que el limite de
                                                        ; X-AnchoSpr?
                        ret     nc                      ; Si, lo es, no actulializamos
                        ld      (Xspr),a                ; Actualizamos la X.
                        if      DOUBLESPRITE
                        ld      (Xspr2),a
                        endif

                        ret


movesprup:              ; C. UP
                        ld      a,(Yspr)
                        sub     1                       ; Y-=1;
                        ret     c                       ; Cy=1. El valor es menor que 0, no
                                                        ; actualizamos
                        ld      (Yspr),a                ; Actualizamos la Y.
                        if      DOUBLESPRITE
                        ld      (Yspr2),a
                        endif

                        ret

movesprdown:            ; C. DOWN
                        ld      a,(Yspr)                ; Y+=1;
                        inc     a
                        cp      192 - 16                ; El valor es mayor que el limite
                                                        ; de X-Alto Spr?
                        ret     nc                      ; Si, no actualizamos
                        ld      (Yspr),a                ; Actualizamos la Y.
```

```
                        if      DOUBLESPRITE
                        ld      (Yspr2),a
                        endif
                        ret



; *** DOS ONLY ROUTINES ****


                        if      DOSMODE


; --------------------------------
; BIOSCALL (DOSMODE)
; Llamada a una rutina
; de la Bios ROM
; IX : Valor de la rutina a llamar
; --------------------------------

bioscall:
                        ld      iy,(EXPTBL-1)
                        call    CALSLT
                        ei
                        ret



; ----------------
; EXIT
; Salida al DOS
; Solo en DOSMODE
; ----------------

exit:

                        ld      a,(saveinitcolors)
                        ld      (FORCLR),a
                        ld      hl,(saveinitcolors+1)
                        ld      (BAKCLR),hl             ; Recuperamos los colores
                        ld      a,(savecliksw)
                        ld      (CLIKSW),a              ; Recuperamos valor clik

                        ld      a,(saveinitscreen)      ; Recuperamos el Screen
                        ld      ix,CHGMOD
                        call    bioscall
                        ld      sp,(savesp)             ; Posicionamos el puntero de pila
                                                        ; con el valor que tenma al entrar
                        ei
                        ret                             ; Y retornamos

                        endif




; ***** DATA ZONE ***

                        ; Datos de nuestro sprite.
sprdata:

                        if      SINGLESPRITE
                        db      00h,00h,00h,06h,18h,27h,1Fh,3Fh
                        db      7Fh,7Dh,7Eh,7Fh,3Bh,3Dh,60h,00h
                        db      00h,00h,60h,0F0h,00h,0FCh,0B4h,0B6h
                        db      0B6h,0FEh,030h,0FCh,0B4h,08Ch,00h,00h
                        else

                        db      00h,00h,00h,02h,08h,07h,1Fh,3Fh
                        db      3Fh,3Dh,3Eh,3Fh,19h,00h,00h,00h
                        db      00h,00h,60h,0F0h,00h,0FCh,0B4h,0B6h
```

```
                        db      02h,0FEh,30h,0FCh,0B4h,84h,00h,00h
                        db      00h,00h,00h,04h,10h,20h,00h,00h
                        db      40h,40h,40h,40h,22h,3Dh,60h,00h
                        db      00h,00h,00h,00h,00h,00h,00h,00h
                        db      0B4h,00h,00h,00h,00h,08h,00h,00h

                        endif



                        if      ROMMODE                 ; Si hacemos una ROM, rellenamos el
fillff:
                        ds      06000h - $,0FFh         ; espacio hasta 8192 bytes, con 0FFh
                        endif



; **** VARS ZONE ***

                        ; Inicio de la zona de memoria de variables
                        ; Varia segun DOSMODE o ROMMODE

                        if      ROMMODE


MEMINIT:                equ     0E000h                  ; Si es Rom, esta es la zona de
                                                        ; variable compatible con un MSX1 de
                                                        ; 8K de Ram
                        else
MEMINIT:                equ     $                       ; Si es DOS, la zona de variables se
                                                        ; aloja en el propio programa, pues
                                                        ; el mismo se ejecuta en RAM.
                        endif


vdp_reg98:              equ     MEMINIT                 ; Variable para port#0 de VDP
vdp_reg99:              equ     vdp_reg98       + 1     ; Variable para port#1 de VDP

Yspr:                   equ     vdp_reg99       + 1     ; Y de nuestro sprite
Xspr:                   equ     Yspr            + 1     ; X de nuestro sprite
patspr:                 equ     Xspr            + 1     ; patron de nuestro sprite
colspr:                 equ     patspr          + 1     ; color de nuestro sprite


                        if      DOUBLESPRITE

Yspr2:                  equ     colspr          + 1
Xspr2:                  equ     Yspr2           + 1
patspr2:                equ     Xspr2           + 1
colspr2:                equ     patspr2         + 1
                        endif


                        if      DOSMODE


                        if      SINGLESPRITE

savesp:                 equ     colspr          + 1     ; Valor de la pila
                        else
savesp:                 equ     colspr2         + 1     ; Valor de la pila
                        endif

saveinitcolors:         equ     savesp          + 2     ; Var. Salva colores
saveinitscreen:         equ     saveinitcolors  + 3     ; Var. Salva Screen
savecliksw:             equ     saveinitscreen  + 1     ; Var. Salva CLIKSW System Var
                        endif

                        end
```

Esta sección reaparece ya que disponemos nuevamente de material para poder comentar. En esta ocasión hemos seleccionado cinco títulos que serán del agrado de los usuarios que siempre buscan en el país del sol naciente una referencia o unos productos exóticos dentro de un mundo cada vez más globalizado. Además comentamos un par de demos para hacer más variado el artículo.

GUNSHOT 4 "vs festa again"

Casa : Rabbit Soft Workers, 2001
Música : FM
Tipo: Habilidad
Sistema: TURBO R (con ratón)

Una vez hemos conseguido dar con la casa Rabbit Soft Workers vemos en su catálogo un juego llamado Gunshot el cual tiene dos versiones, la primera titulada *Gunshot 4 Vs Festival* del año 2000, y la segunda *Gunshot Vs Festa Again* del año siguiente que es la que comentaremos a continuación. La diferencia entre ambas es básicamente la calidad gráfica, en que la segunda parte mejora notablemente a su antecesora. En el catálogo aparece otro título más novedoso llamado *Gunnership 2 On 2 Heroines Edition* pero debido a que no hemos podido hacerlo funcionar ya que no se encuentran algunos archivos que requiere el comando ejecutable. Una verdadera lástima, pero si en algún momento consiguiéramos hacerlo lo haríamos comunicar.

Como es habitual el juego se nos presenta en idioma nipón lo cual hace un poco complicado comprender como es debido las normas del mismo. Podemos jugar en disco (funciona en el emulador de Ascii excepto la emulación del ratón) o también desde disco duro (recomendable). Tras la carga inicial que se nos presenta con una barra de Now Loading finalmente carga y se nos presenta una intro simple con un pequeño scroll de un dibujo de una de las protas (destacar el cambio de paletas) que tiene una efímera animación.

En sí el juego no es más que a ver





quién consigue acertar disparando más veces moviendo el ratón a los iconos que irán apareciendo tipo Pico-Pico. Lo cierto es que hay fases variadas y modos de juego distintos lo que lo convierte en más ameno para el jugador.

Gráficamente el juego en sí es sencillo y casi que de lo que más podemos hablar es de un diseño muy detallista y trabajado, colorista a más no poder y sobretodo muy definido. La verdad es que pocas veces vemos un nivel tan alto en diseño de pantallas. Todo es preciosista, hasta la tabla de ranking de records. Destacamos el *How To Play* que nos recuerda enormemente a las producciones de Snk para Neo Geo.

http://www9.ocn.ne.jp/~rabbits/

DEAD OR LIVE

Casa : Ctrl+Break, 1997
Música : FM
Tipo: Lucha
Sistema: TURBO R

La primera impresión nos recuerda enormemente al Brain Drive ya comentado en esta sección hace dos números. Y es que parece que los juegos de lucha causaron furor entre nuestros usuarios y trataron de llevar a cabo diversos proyectos que parece que nunca llegaron a buen puerto, ya que no se acabaron completamente, se quedaron a medias y este Dead Or Live no es una excepción. Todavía se quedó más corto que el Brain Drive. La verdad es que es curiosa la historia de los juegos de lucha post Street Fighter. En el último número de la Msx Fan veíamos una promo para Turbo R, Gekikara, con gráficos super pixelizados pero que tenía una excelente animación. Este juego llegó a una fase de acabado muy aceptable en 1995 donde Delta Trial, el grupo creador, abandonó definitivamente el proyecto y dejó el listón muy alto. Esto dio pie a que en nuestro sistema se podían crear este tipo de juegos, ya que en consolas similares de 8 bits habían versiones muy jugables de los grandes clásicos de la lucha. De esta forma tuvimos un Street Neo Fighter que pese a ser un intento trabajado tampoco consiguió ser un juego de culto del cual presumir. El Be Bop Bout pese a su excelente calidad no era una juego de lucha sino de cartas y el Street Bison 3 fue otro intento chapucero de sprites monocromos. En cambio Brain Drive sí que tenía una pinta genial. De haber estado acabado sería la joya del Msx en juegos de lucha.

Después de habernos puesto al día en este poco productivo género vamos a analizar un poco lo que ha dado de sí este proyecto truncado llamado Dead Or Live.

En primer lugar destacar una pantalla de selección de personajes muy elaborada, con unos gráficos estupen-





dos. A destacar el mapamundi y las caras de los personajes. Hay que decir que sólo podemos jugar contra un segundo jugador ya que la opción CPU no existe, una lástima si no tienes con quien probarlo.

Parece que sólo podemos luchar en un par de escenarios, los cuales son muy coloristas pero se echa de menos alguna pequeña animación, en la cascada de agua por ejemplo. Lo que son las barras de energía y marcadores están muy logrados. Los gráficos de los personajes están muy bien animados y disponen de varios ataques tipo Street Fighter. Una combinación de un cuarto de curva en el mando y pulsando puñetazo o patada veremos cómo surgen. Algunos son bastante espectaculares. Como punto negativo vemos que los personajes parpadean en exceso, como si le faltaran frames.

La verdad es que el juego no da para más, se quedó en un fase de pruebas y gracias que es un poco jugable, pero demuestra que nuestro ordenador es capaz de disponer de juegos de lucha decentes, siempre que hubiera un grupo desarrollador entusiasta, ya que estos juegos requieren mucho esfuerzo y tiempo para llevarse a cabo, por este motivo se quedan en agua de borrajas en casi todos los casos.

COSMOGANG DE PUZZLE

Casa : Ctrl+Break
Música : FM
Tipo: Puzzle
Sistema: MSX2

Estamos ante otro juego de aquellos que podrían pasar por clones del famoso Tetris o Columns, pero quizá se parezca más al no tan conocido Pikiinya de Super Nintendo aunque mucho menos elaborado.

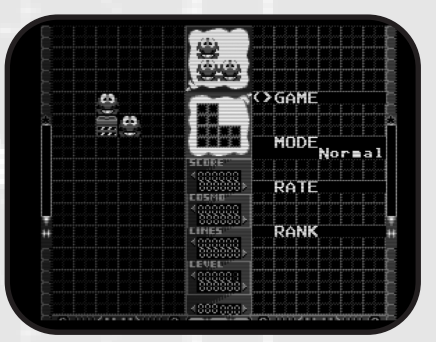
Una vez cargado el programa aparece el logo animado de sus creadores (muy logrado por cierto) y pasamos a la pantalla de título en screen 5 muy colorista y con una melodía en FM bastante aceptable. Pulsamos y llega-mos a la pantalla de juego donde podemos seleccionar el modo de dificultad del juego. La verdad es que incluso el más sencillo resulta un poco complicado.

Podemos ver que pueden jugar hasta dos jugadores simultáneos. La dinámica del juego es sencilla. Mientras caen figuras de tres piezas (las podemos ir rotando) debemos hacer líneas con las cajas, pero unos caretos nos irán complicando la vida. La única manera de quitárnoslos de encima será con el icono circular que lleva una flecha de dirección indicando por donde arrasará cabezas a su paso. Así de sencillo pero así de dificultoso.

En definitiva, un programa que podía estar un poco más elaborado, no sé, quizá la inclusión de algún icono más para hacerlo más variado. Quizá se salva la melodía de la pantalla de juego, sobretodo si prestamos atención al bajo. Si no lo conocías es una excusa para descargarlo y probarlo, no sé quizás os parezca estupendo.

http://www.yashok.com/~yashok/msx/ctb/ctb_index.shtml





MR HAWAII (Save the Earth)

Casa : Rats, 1991
Música : PSG
Tipo: Matamarcianos
Sistema: MSX2 (o superior)

Otro de esos juegos japoneses difíciles de encontrar por la red. Pese a tener ya unos cuantos añitos, es la primera vez que damos con él, y la verdad es que nos hemos llevado una grata sorpresa porque nos encontramos ante un juego de naves de bella factura.

Es cierto, no estamos ante un prodigio, pero tenemos un juego que pese a presentarse en una pequeña ventana disponemos de una jugabilidad maravillosa. Lo más parecido que se nos puede pasar por la cabeza sería el mítico Barumba. Jugamos con teclado o

joystick y con la barra espaciadora disparamos de frente y con Shift nuestra nave gira y dispara hacia atrás, así de curioso; o en su defecto Z y X serán las teclas que suplen a las anteriormente mencionadas.







Los gráficos son muy buenos, sobretodo los fondos y los enemigos que son sencillos pero divertidos (destacamos los final bosses).Y si nos fijamos bien en la fase tres verremos un triple scroll parallax para quitarse el sombrero. La música no apuesta por el FM y se queda en simple sonido PSG, aunque no desmerece, son melodías interesantes. La de la presentación nos recuerda a una mítica canción del Psycho World.

Echamos de menos la puntuación, ya que en un tipo de juego como éste nos gusta saber cuántos enemigos somos capaces de derribar. La nave dispone de una barra de protección (shield), que nos indicará los impactos que faltan para que acabemos la partida. Durante ésta podremos ir aumentando puntos del escudo si recogemos buenos items que aparecerán en cofres. ólo hay dos niveles de dificultad, y lo malo es que no hay un bello final, sólo un simple congratulations.

Por supuesto debemos ejecutar el TurboBasic antes de cargar el programa, sinó no funcionará, ya que no viene entre los ficheros el Msxkun.

CELL 4

Casa : Rats, 199X
Música : PSG
Tipo: Habilidad
Sistema: MSX2



Otro juego de la misma casa Rats pero que no sabemos en qué año se elaboró, si antes o después del Hawaii. Lo que sí sabemos es que es un juego muy bien hecho dentro de lo sencillo de su desarrollo. Es un juego de habilidad que en MSX podemos recordar alguno similar, quizá el Artic sea el más parecido.



Tenemos una esfera que nada más pulsar la barra espaciadora coge inercia y debemos ir controlándola por el mapeado hasta llegar a un cuadrado donde aparece la palabra “IN”. El sistema de juego es muy simple, pero poco a poco el llegar a la meta se nos va a ir complicando, ya que cada fase que avanzamos se nos muestra más difícil el camino. Entre los obstáculos más incordiosos tenemos el agua, que su simple contacto nos restará una vida. Hay otras esferas que se van moviendo y nos dan problemas para pasar. También hay puentes móviles para cruzar el agua. Los muros nos harán rebotar; este efecto está muy bien logrado, quizá sea lo mejor del juego.



Los gráficos son coloristas con tonos apastelados, bien trabajados y con unas sombras sutiles que dan sensación de profundidad. Gráficamente da una sensación immejorable. También aquí la pantalla de juego es una ventana, no ocupa toda la pantalla. La presentación está bien, con una imagen fija que emula un entorno tridimensional, mientras que las letras del título hacen un efecto rotatorio que solemos ver con frecuencia en muchas demos.

En cuanto al sonido, pues igual que el Hawaii, la música es en PSG. Y digo la música ya que tan sólo hay un único tema en todo el juego. Podemos decir que es un tema bastante animadete para darle emoción a la partida. Parece que tiene un ritmo que recuerda a las primeras partituras del rock más añejo.

http://riy.jp/200401.html

IMPORTED 2004-2005

Casa : VV.AA., 2005
Música : PSG
Tipo: Demo
Sistema: MSX

En los ochenta eran típicas las conversiones de Spectrum a MSX, eso sí, siempre juegos (ciertamente tenemos muy malos recuerdos de ese mal hábito). En cambio demos creo que no era muy habitual. En este caso disponemos de una demo que previamente fue creada para los ZX Spectrum y que ha sido convertida a la primera generación de MSX. Y pese a su sencillez nos ha hecho gracia dicha conversión.



No estamos ante una demos elaborada, sinó todo lo contrario. Es una sucesión de imágenes fijas con su propia melodía. Van pasando a medida que pulsamos la barra espaciadora. Así de sencillo, hasta que llegamos a los créditos.

Por el contrario tenemos que admitir que los gráficos son estupendos, exceptuando uno o dos, todos realizados por dibujantes diferentes y que muestran un dominio técnico muy bueno pese a la baja resolución y la escasez de color.



La música casi que lo mismo que los gráficos, cada pantalla tiene su propia melodía en PSG, con ese sonido característico de otros ordenadores no MSX, y que volvieron a poner de moda el grupo Matra hace unos añitos. Para todos los gustos. Algunas son versiones de clásicos como la melodía de la serie Dallas o Englishman in New York.

MSX UNLEASHED

Casa : Daniel Vik and Vincent van Dam, 2006
Música : PSG
Tipo: Demo
Sistema: MSX



¡Brutal! Estamos ante una de las mejores demos creadas jamás para MSX1. Esta demo crea efectos que no hubiéramos creído posible en los MSX de primera generación.

Daniel Vik ya nos daba un anticipo de lo que era capaz de programar para MSX1 en su demo Waves, pero en esta ocasión, junto a Van Dam se han superado con creces.

En la pantalla de título unas franjas de colores avanzan rodeando al texto tanto por delante como por detrás, algo que sólo habíamos visto antes en un MSX2.

La siguiente carga nos ofrece en el plano final el scroll de un texto de tamaño enorme mientras nos aparece en medio de la pantalla por delante una amplia franja donde vuelve a aparecer más texto siguiendo la forma de onda.

Más adelante vemos gráficos fotográficos que nos ofrecen 105 colores simultáneos. Parece inspirado en el efecto que ya consiguió NOP en su mítica Unknown Reality para MSX2. Parece asombroso ver tantos colores en un simple MSX1, pero es cierto.



Continúa la demo con un scroll de fondo cuadriculado sobre el cual otro texto va moviéndose por la pantalla rotando sobre sí mismo.

Y la última parte es quizá la más espectacular. Aquí ya vemos imágenes creadas por caracteres que rotan, se alejan y se acercan (efecto zoom). La verdad es que es impresionante.



En los créditos aparece una esfera en tres dimensiones sin opacidades y creada a modo de puntos que va dando botes en la pantalla.

A todo esto debemos mencionar la música, que ante tanto efecto visual parece que queda en un segundo plano. No es que sea increíble, pero es muy correcta, de estilo techno.

http://es.msx.org/



HINOTORI

Contraseñas

Para introducir las contraseñas hay que pausar el juego con **"F-1"** y seguidamente pulsar **"HOME"** para visualizar la contraseña actual; para introducir cualquier otra hay que volver a pulsar **"HOME"**.

Una vez tecleada la contraseña, hay que pulsar la tecla **"RETURN"** para confirmar.

FULLITEMDAYOON

Obtenemos una unidad de cada objeto, incluida la brújula.



METALSLAVE

Obtenemos 200 plumas de fuego. Una vez.



SUPERBALL

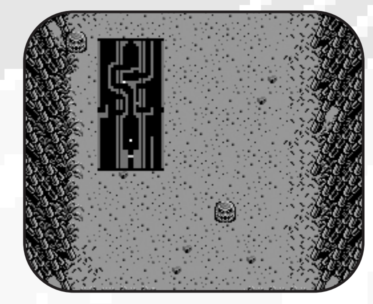
Obtenemos todos los símbolos esféricos.



KOKOWADOKO

Obtenemos seis pergaminos que nos permiten consultar el mapa de la zona seis veces. Podemos consultarlo pulsando **"F-4"**.





TURBO

Obtendremos tres zapatillas, por lo que nuestro personaje se moverá mucho más rápido.



HOIHOIHOINOHOI

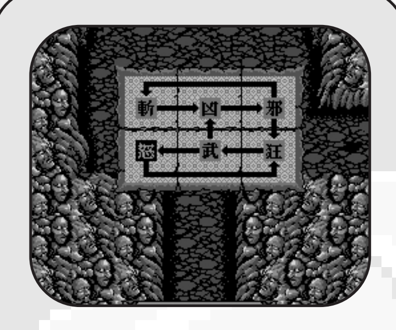
Obtenemos la brújula. Que nos permite saltar de nivel, si disponemos de los pergaminos adecuados.



DOKODEMOMAP

Obtenemos todos los pergaminos, 6 en total, del mapa que engloba todos los niveles del juego. Si disponemos de la brújula, podremos seleccionar el nivel y saltar directamente hacia él.





ILOVEHINOTORI

Inmunidad ante enemigos y disparos pero podemos ser aplastados por las paredes.

ULTRABOX

Obtenemos todos los regalos.



AUTOSHOT

Obtenemos la máxima velocidad de nuestro disparo. Ademas podemos dejar la barra espaciadora pulsada y nuestro personaje disparará automáticamente.



KINOOOIHITODANE

Obtenemos todos los sellos que nos permiten abrir todas las puertas.



HAYATE

Obtenemos la máxima velociad de nuestro disparo.

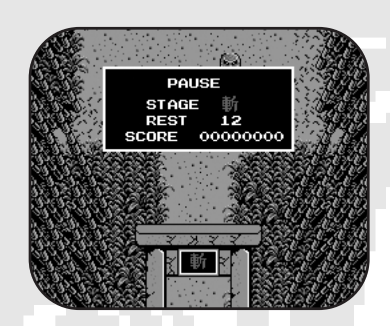
HANEYOKAGAYAKE

Las plumas que obtenemos siempre son brillante, de valor equivalente a 10 plumas.



NANDANANDANANDA
Vidas infinintas.

GAOOOOOOOOOOH
Obtenemos 10 vidas extras.



ENDDEMOGAMITAINA
Pasaremos a ver la secuencia final del juego.

Combinaciones

Si colocamos el QBERT en el SLOT 2, aparecerá un menú secreto, al igual que ocurre cuando se inserta el GAME MASTER. En la pantalla de título, cuando presionemos la barra espaciadora para comenzar la partida, aparecerá un menú que nos permitirá escoger la pantalla en la cual queremos empezar y el número de vidas disponibles.



KING'S VALLEY 2



Contraseñas

Podemos acceder a cualquier pantalla mediante la siguiente formula: se trata de buscar en la tabla inferior las letras que coinciden en la coordenada horizontal y vertical de la pantalla a la cual queremos acceder; escribimos primero el número o letra de la fila y después la letra de la columna, y finalmente añadiremos la palabra **KONAMI**.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	-	-	-

Por ejemplo, para acceder al nivel 50 hay que introducir **3CKONAMI** como contraseña. Comenzaremos la partida con 47 vidas disponibles.

Contraseñas especiales

FESTIVAL
Inmunidad total.

TRYAGAIN
Activamos la posibilidad de continuar cuando la partida termine.



Test musical

Podemos acceder a una zona secreta, donde se encuentra un pequeño piano, que nos permite escuchar las melodías del juego.Hay varios accesos a esta zona situados en seis pantallas diferentes.

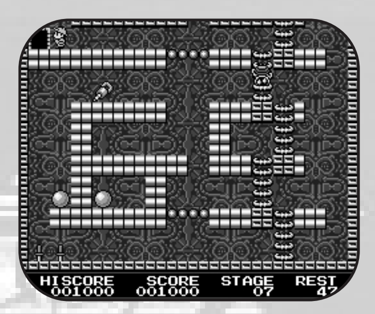
Para descubrir las entradas a dichas zonas primeramente hay que saltar



sobre la zona indicada y aparecerá una pirámide. Si esperamos un poco sobre la piramide descubriremos una puerta que nos permitirá la entrada a la zona secreta si pulsamos el cursor direccional de abajo.

Nivel 7

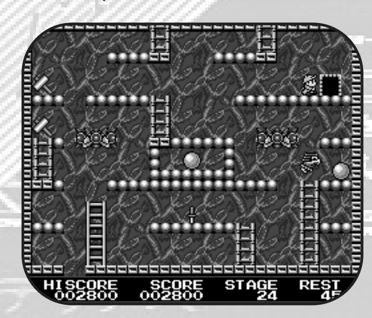
Acceso situado en la parte superior izquierda de la pantalla.



KING'S VALLEY 2

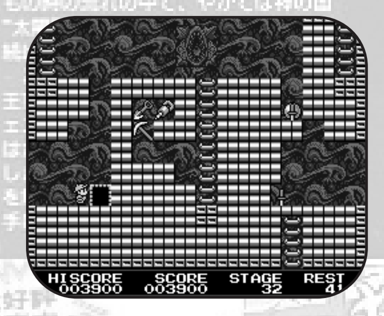
Nivel 24

El la pantalla situada más al sur, en la esquina superior derecha, encontraremos la pirámide secreta.



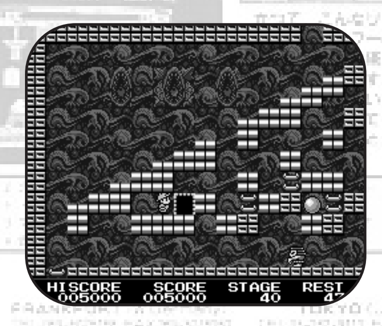
Nivel 32

Acceso situado en la pantalla inferior, en la esquina inferior derecha aproximadamente.



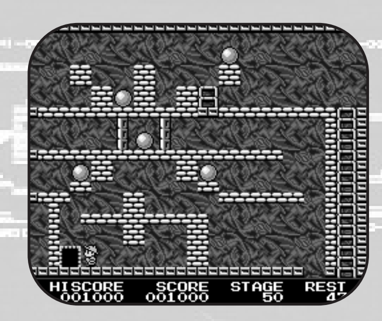
Nivel 40

Acceso que se encuentra en la pantalla superior izquierda; situado horizontalmente hacia el centro y verticalmente cerca de la parte inferior de la misma.



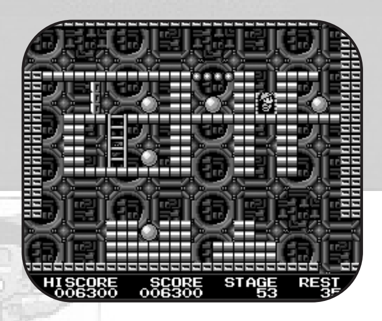
Nivel 50

Situado en la pantalla superior derecha; en la esquina inferior izquierda de la misma.



Nivel 53

En la pantalla que se encuentra más a la izquierda, en la esquina superior derecha, en el hueco que hay a la derecha del pequeño puente, hallaremos el acceso al test musical.



Combinaciones

Con **Game master 2** en Slot 2 podemos acceder a una pantalla secreta que nos revela parte de las contraseñas de inmunidad y de continuar el juego al finalizar la partida.

Para ver esta pantalla hay que mantener pulsada la tecla **"["** y empezar una partida seleccionando *GAME* y pulsando la barra espaciadora.

MAD MIX GAME



Para obtener vidas infinitas hay que esperar a que aparezca la pantalla con los nombres de los programadores y justo en ese momento pulsar simultáneamente las teclas **"ESC"** y **"STOP"**.

Después, comenzaremos el juego y nuestras vidas nuncá disminuirán.

NINJA (DEXTER)

Test musical

Para acceder al menú musical hay que mantener la tecla **"F"** pulsada mientras cargamos el juego.

Seleccionamos los temas con las teclas direccionales y lo escuchamos pulsando la tecla **"SHIFT"**.

MUSIC SELECT

▶ STAGE1 BOSS1
STAGE2 BOSS2
STAGE3 BOSS3
STAGE4 BOSS4
STAGE5 BOSS5
STAGE6 TITLE
STAGE7 ENDING
STAGE8

También es posible escuchar los diferentes efectos sonoros del juego si vamos pulsando la tecla **"RETURN"** en la pantalla de elección del personaje.



METAL GEAR

Códigos

Durante el juego podemos recurrir a una serie de códigos secretos que nos facilitaran mucho el juego. Para introducir dichos códigos hay que pausar el juego con **"F-1"**, seguidamente escribiremos el código y después volveremos a pulsar **"F-1"**.

Hay que respetar los espacios que se encuentran en los códigos y que estan representados con el caracter "_".

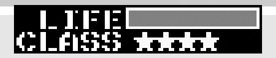
DS_4

Aumentaremos una graduación y con el correspondiente aumento de vitalidad.



ANTA_WA_ERAI

Obtendremos las cuatro graduaciones posibles obteniendo así la máxima vitalidad.



HIRAKE_GOMA

Obtendremos todas las tarjetas, en total 8 que nos permiten abrir cada una de las puertas del juego.



ISOLATION

Podemos coger hasta 999 raciones de alimentos.



INTRUDER

Podemos recoger munición de cualquier tipo hasta llegar a las 999 unidades.



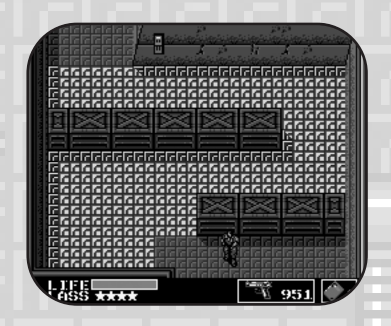
Derrota a Metal Gear

Para acabar con Metal Gear tenemos que colocarle 16 bombas plásticas. La secuencia nos indica en que pie debemos hacer explotar cada bomba. El pie izquierdo se representa con la "I" y el derecho con la "D".
"D", "D", "I", "D", "I", "I", "D", "I", "I", "D", "D", "I", "D", "I", "D", "D"



Inmune a la electricidad

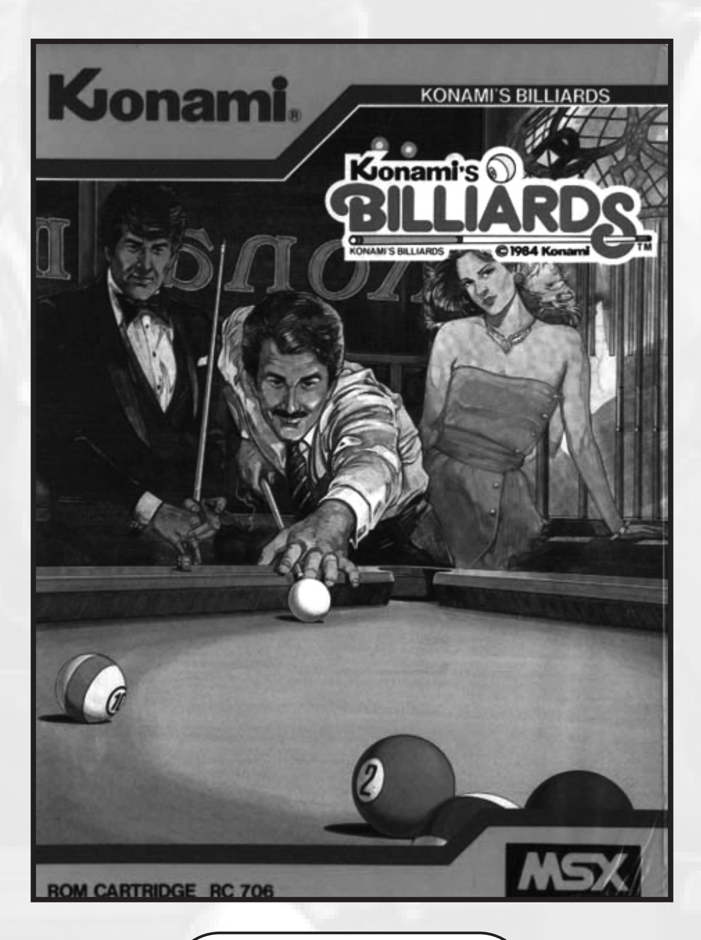
Utilizando el mismo procedimiento que se utiliza para el truco de los objetos, podemos atravesar una zona electrificada sin recibir daños pulsando seguidamente **"F-1"**, **"F-4"** ó **"F-5"**, **"ESC"** y **"F-1"**.



Tiempo extra para escapar

Podemos obtener 2000 unidades de tiempo extra para escapara de *Outen Heaven,* si seleccionamos los cigarrillos para que nuestro héroe se relaje.





RC706
Konami's Billiards
Video Hustler (Japón)
Konami 1983
8kb ROM

Konami's Billiards es un sencillo juego de billar americano con el que podemos echar unas partidas en solitario o contra nuestro querido MSX.

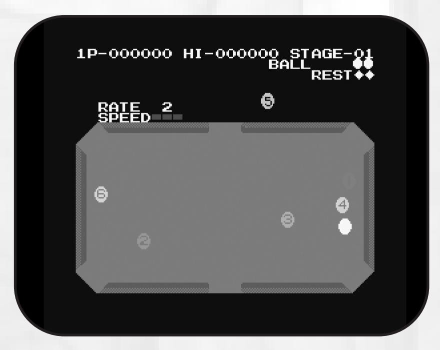
El sistema de juego

Mediante los cursores podemos seleccionar la dirección hacia la cual queremos dirigir la bola. Una vez ajustada la trayectoria, prestaremos atención a la barra de velocidad de tiro,



que se incrementa y decrementa automáticamente; pulsaremos la barra espaciadora justo cuando se sitúe en la potencia que deseamos dar a nuestra bola y ésta saldrá disparada hacia la dirección seleccionada.

En cada tirada tenemos tres oportunidades de meter una bola (REST). Si no conseguimos meter ninguna, perderemos una bola (BALL); si esto sucede tres veces, la partida habrá acabado.

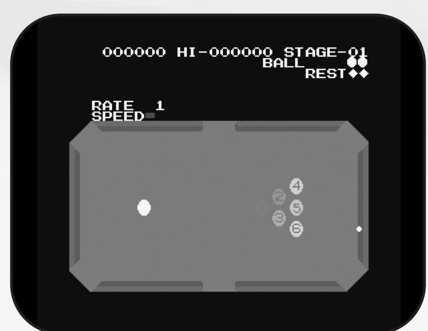


La puntuación

El sistema de puntación funciona mediante la siguiente fórmula:

nº de bola x bonificación x bolas metidas a la vez o consecutivamente (este valor se duplica por cada bola introducida -RATE-)

Si todas las bolas son introducidas se pasará a la siguiente pantalla y se volverá a "romper" de nuevo.



Gráficos 1 2 3
Sonido 1
Adicción 1 2 3 4 5
Originalidad 1 2 3

Mapa fotográfico de THE CURE

El nuevo juego para MSX1 del desconocido grupo ***X2LS*** se ha llevado el primer premio de la pasada *MSXdev'05* y es que, después de probarlo, podemos afirmar que se lo ha bien merecido.

Si este grupo mantiene la misma calidad con sus próximos juegos, como el espectacular "Dragon" podemos estar seguro que disfrutaremos de unos increíbles títulos en nuestros MSX.

El juego se divide en cinco niveles, cada uno de ellos con dos zonas. El objetivo principal es encontrar una llave maestra que se encuentra oculta en cada zona y que nos permite avanzar hasta la siguiente.

Nivel 1 - Entrada al castillo

Para acceder al castillo debemos cruzar un puente en ruinas, así que tenemos que vigilar donde ponemos los pies y estar alerta porque criaturas del pantano saltarán sobre nosotros. Las cortinas y las vidrieras del castillo nos son familiares, pero no hay que distraerse y buscar la llave maestra, que se encuentra escondida en el interior de los muros de piedra. El guardián de este nivel es una gigantesca alma en pena.

Nivel 2 - Catacumbas

Un lugar infestado de peligrosas arañas gigantes, así que hay que avanzar con cuidado. También se debe vigilar mucho con los saltos porque fácilmente podemos precipitarnos al vacío. La propia muerte con su enorme guadaña nos espera al final del nivel.

Nivel 3 - Cascadas

Avanzaremos entre cascadas y realizaremos peligrosos saltos. El agua oculta preciados objetos pero también sirve de morada para una criatura feroz, una serpiente de las profundidades; detener el tiempo puede ser útil para derrotarla.

Nivel 4 - Las celdas

Este lugar ha sido la prisión para muchas almas que, con ganas de venganza, poseen los huesos esparcidos por todo el lugar y forman esqueletos que nos atacarán sin piedad. Un gran cráneo es en este caso el guardián.

Nivel 5 - La torre del mal

Desde la torre vemos la luna teñida en sangre; es la torre donde se encuentra el mismisimo señor del mal, plagada de trampas y falsas llaves. Hay que buscar bien para encontrar el camino que nos lleva a la batalla final.



Objetos

LLave
Nos permite abrir un cofre.

Bastón
Nos permite abrir cuatro cofres.

Llave maestra
Abre la puerta que da paso a otra zona o al guardián.

Anillo del poder
Destruimos a todos los enemigos solo con tocarlos.

Lágrima
Somos inmunes, durante unos segundos.

Bola de energía
Recuperamos nuestra vitalidad.

Látigo
Incrementamos el poder y alcance de nuestro látigo.

Espada de fuego
Disparamos ráfagas de fuego. - Arriba + disparo -

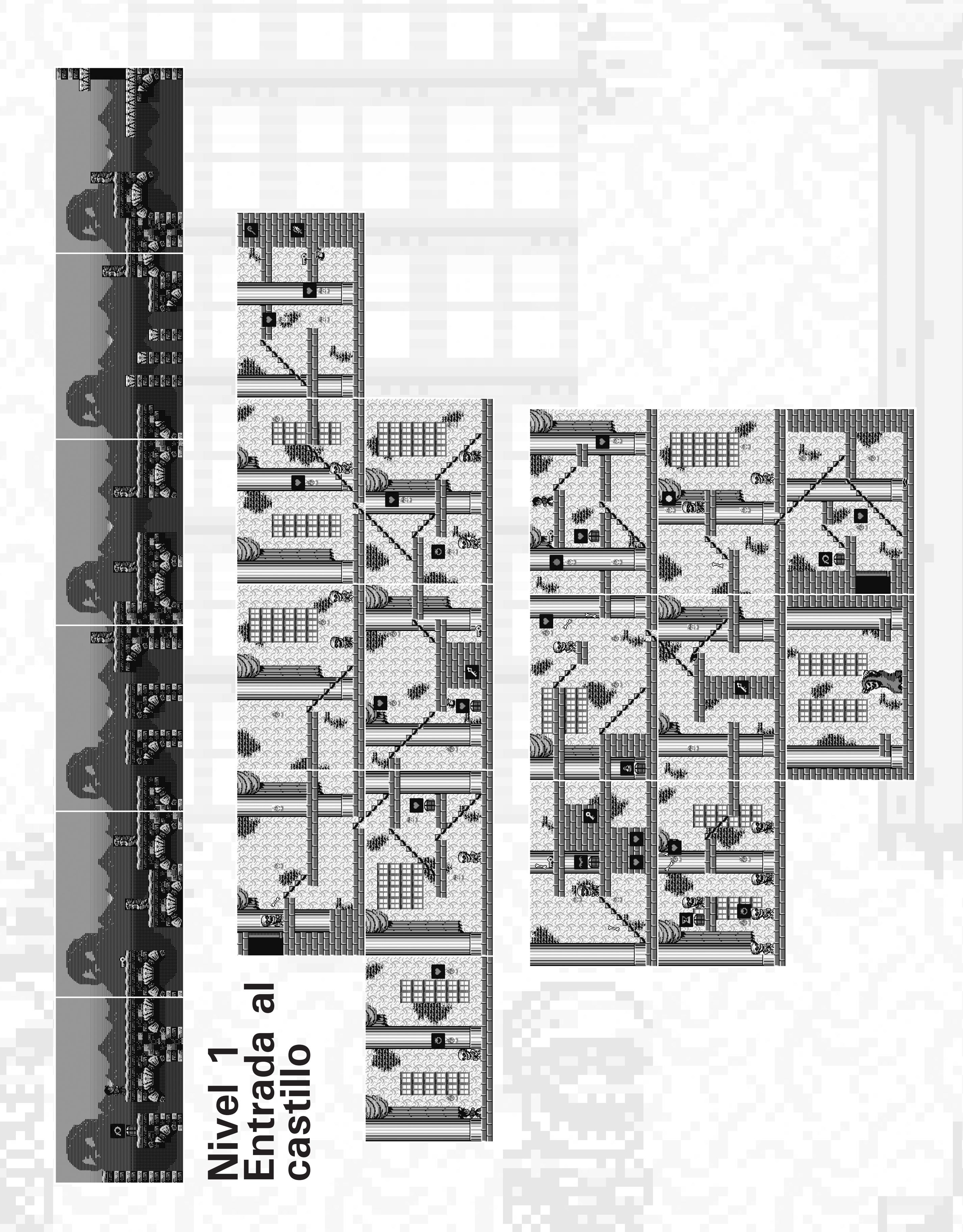
Pócima - Lanzamos pócimas que rompen muros. - Arriba + disparo -

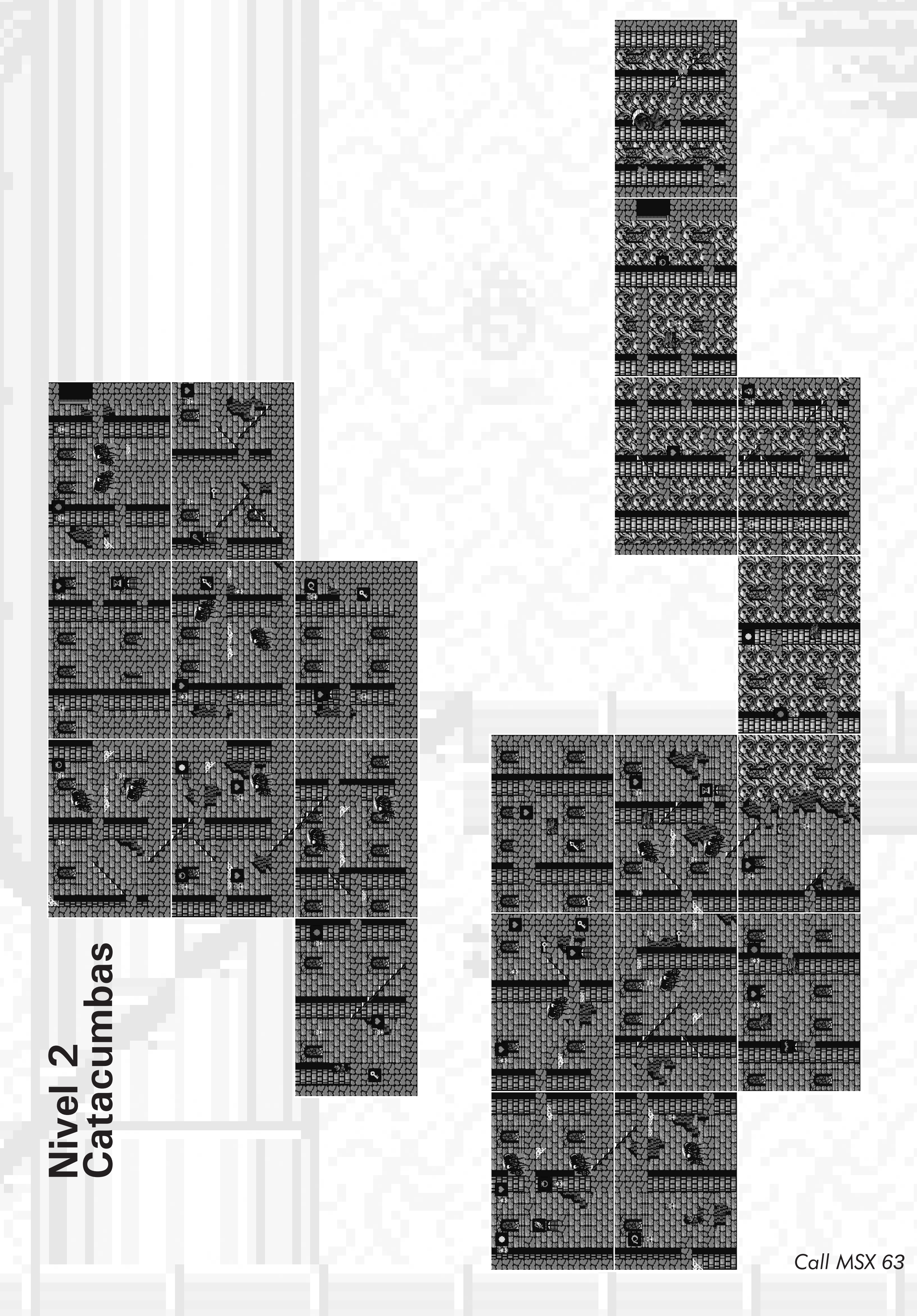
Reloj de arena
Detenemos la acción unos segundos. - Abajo + salto -

Murciélago
Obtenemos un crédito para continuar la partida.

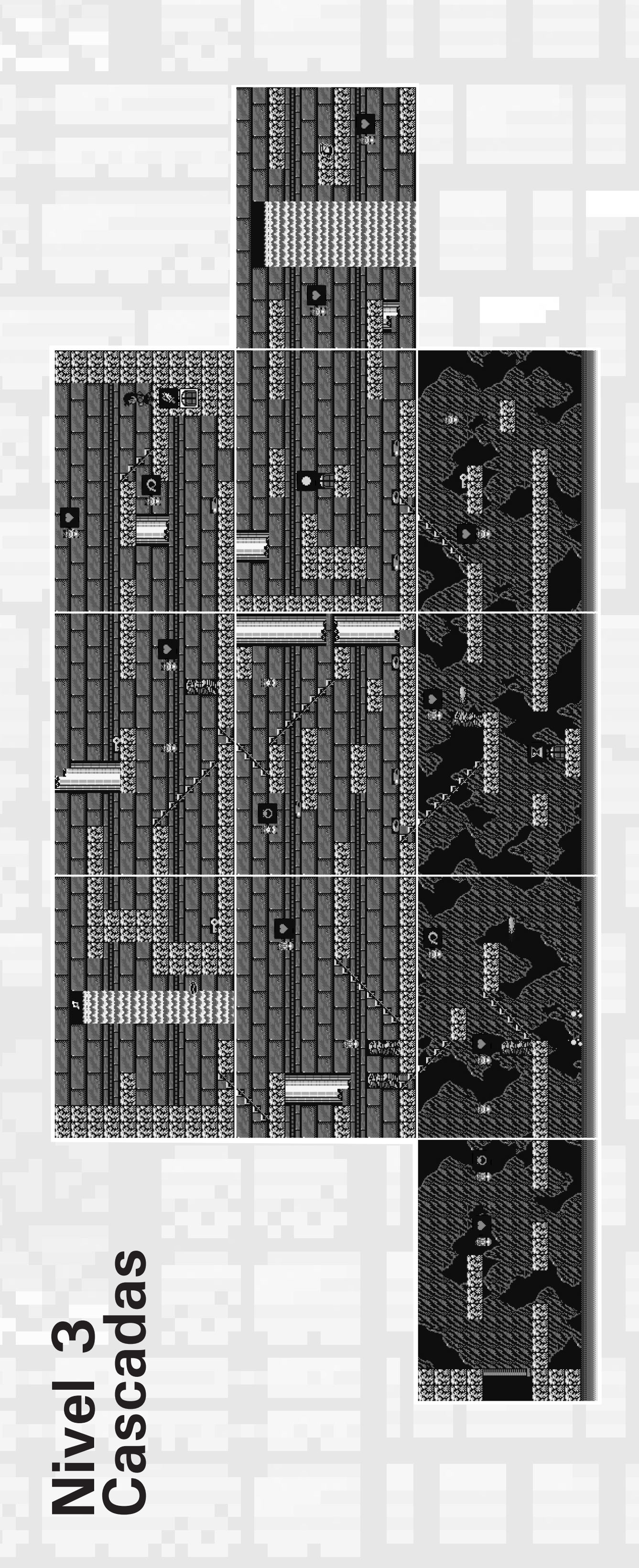
Corazón
Obtenemos cinco corazones.

Mapa fotográfico

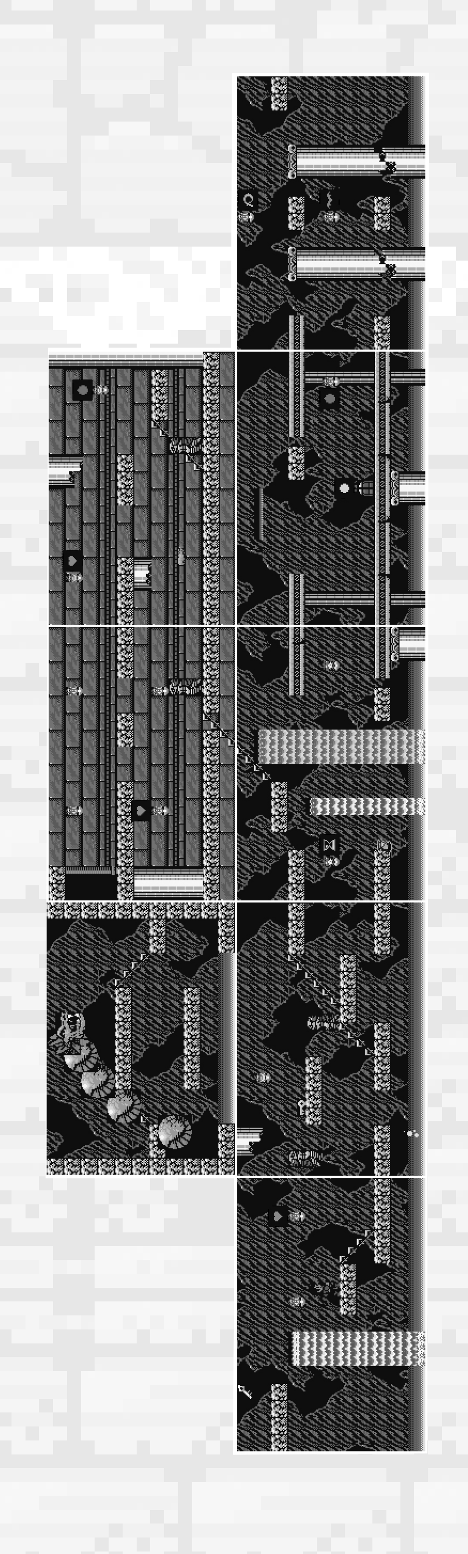


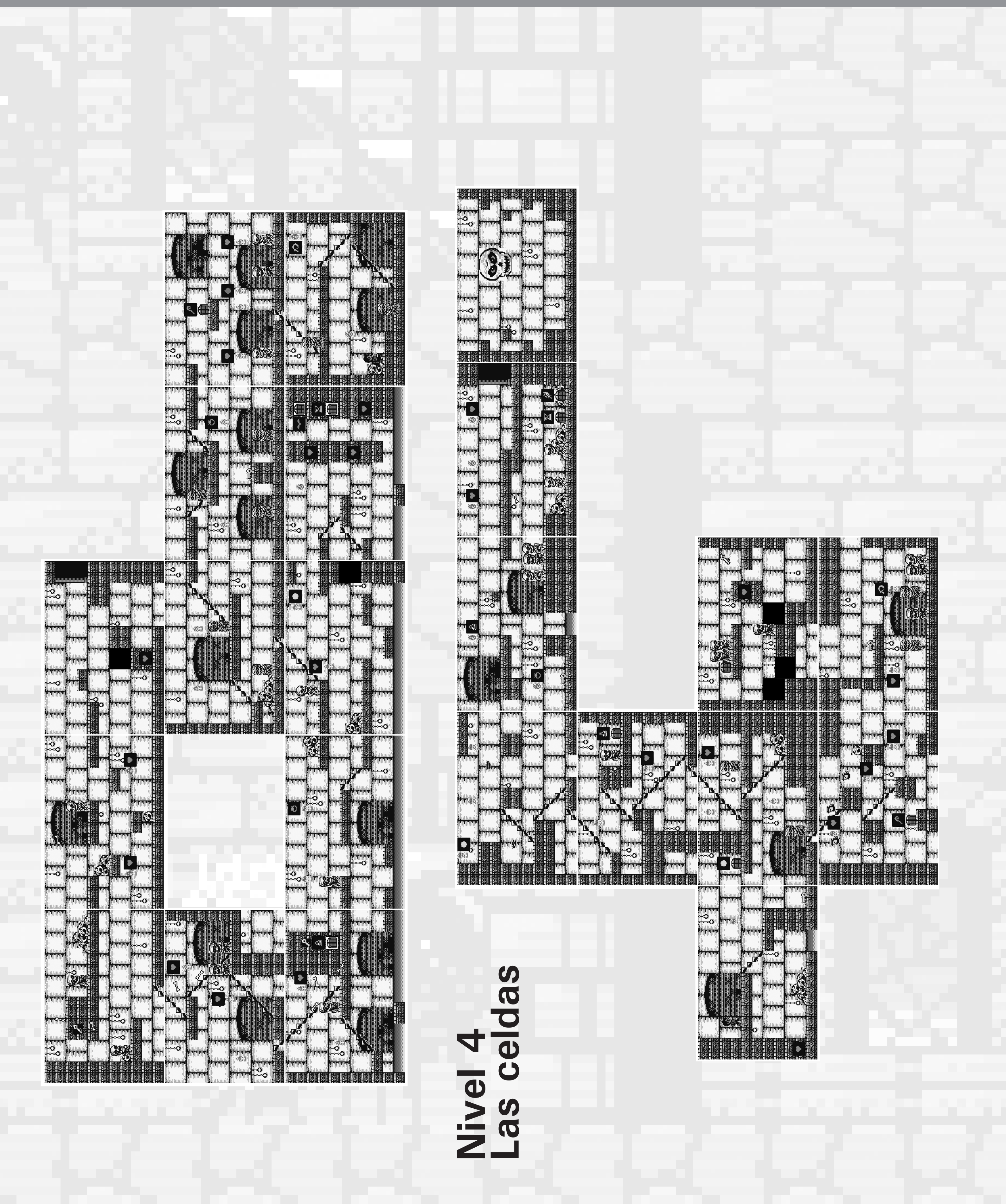

Nivel 2
Catacumbas

Mapa fotográfico

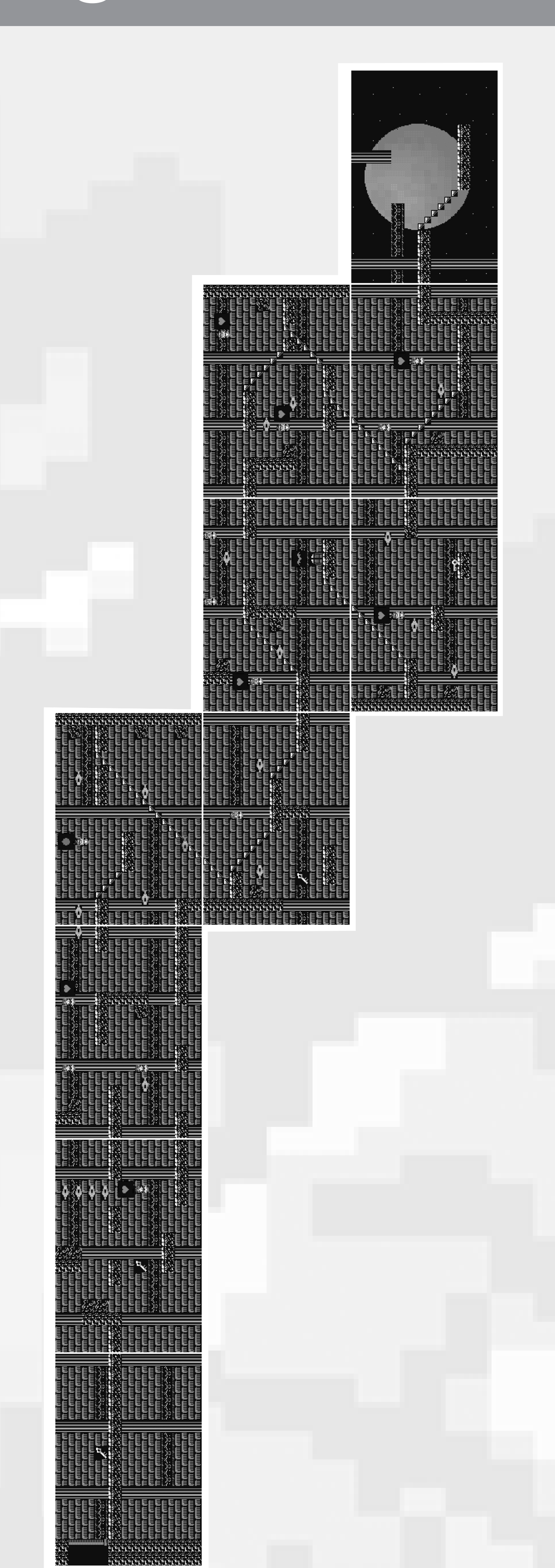


Nivel 3 Cascadas

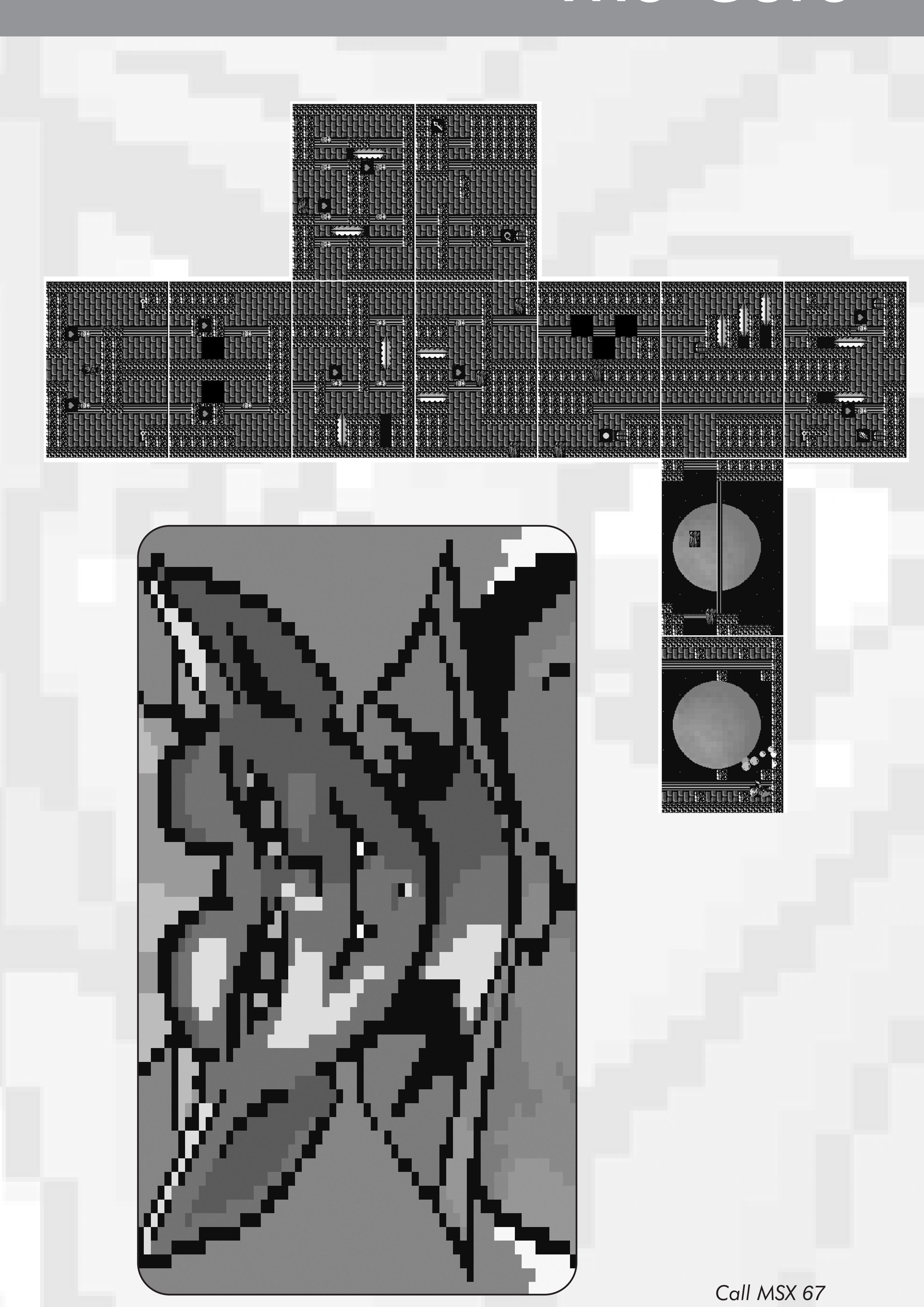



Nivel 4
Las celdas

Mapa fotográfico



Nivel 5
La torre del mal





WWW.
MATRANET
.NET
SCHWEPPES
HOT
RYP CAPITOL
propia vía.
HOTEL CAPITOL
tryp
Banco Atlántico
PRESS S★T★A★R TO CONTINUE
TODAY THE ARCADE - TOMORROW THE WORLD